CS/COE 1520

pitt.edu/~ach54/cs1520

Regular expressions

Regular expressions

- Formally:
  - Expressions that can be generated by regular languages, or that can be produced by a finite automaton
- Practically speaking:
  - Patterns that you can use to match various parts of strings, allowing matches to be made when the exact values to be matched are uncertain
    - E.g.,
      - Find where email addresses appear in a string of text
      - Check if a string represents a valid phone number

Use in Javascript

- Will primarily use regular expressions with 4 Javascript string functions:
    - search()
    - match()
    - replace()
    - split()

Find pattern instances in string

Replace instances of pattern with other text

Break up the string using pattern as a boundary

Defining regular expressions

- Two approaches in Javascript:
    - `new RegExp(`*`pattern`*`[, `*`flags`*`]);`
        - E.g., var re = new RegExp("snipe");
    - */pattern/flags*`;`
        - E.g., `var re = /snipe/;`

Those were very boring examples

- `/s*n[iI1]p[eE3]/` will match the following:
  - snipe
  - sssnipe
  - ssssssssssssnIp3
  - sn1p3
  - nIpE

What was the * indicating?

- This is one of the indicators for matching repeated characters (or classes or patterns)
    - *
        - Repeated 0 or more times
    - +
        - Repeated 1 or more times
    - ?
        - Occurs 0 or 1 times
    - {n}
        - Repeated exactly n times
    - {n, m}
        - Repeated between n and m times

OK, and the []?

- [] allows for the creation of *character sets*
    - E.g., [il1] matches:
        - i
        - l
        - 1
    - It does not match:
        - l1
        - iii
        - 1i

How could we match these?

Complement character sets

- If a ^ appears as the first character in a character set, that set will match any character *not* listed in the character set.
  - [^il1] matches:
    - q
    - 7
    - T
  - [^il1] does not match:
    - i
    - l
    - 1

More character sets

- [abcdefghijklmnopqrstuvwxyz]
- [a-z]
  - What would happen: `"A".search(/[a-z]/)`
- [A-Za-z0-9]
- [^A-Za-z0-9]
- [aeiouAEIOU]
- [0-9+-\/*]
  - What does this match?

Builtin character sets

- \d
  - Digits
  - = [0-9]
- \D
  - = [^0-9]
- \w
  - "Word" characters, or any alphanumeric character
  - = [A-Za-z0-9_]
- \W
  - = [^A-Za-z0-9_]
- \s
  - "Space" characters (e.g., space, tab newline, etc.)
  - =[\f\n\r\t\v\u00a0\u1680\u180e\u2000-\u200a\u2028\u2029\u202f\u205f\u3000\ufeff]
- \S
  - Non-whitespace characters
- .
  - Any character

Anchors

- ^
  - Matches the beginning of a string
  - Unless in multiline mode, then matches the beginning of a line
- $
  - Matches the end of a string
  - Unless in multiline mode, then matches the end of a line
- \b
  - Word boundary
- \B
  - Not a word boundary

Greedy vs Lazy evaluation

- By default matches are greedy from left
  - If multiple characters can be matched, as many are consumed as possible left to right, as long as overall match can still succeed
- Backtracking may be needed to obtain overall match
- We can change the matching to be lazy by putting a ? after the repetition operator
  - E.g., /a*?/
    - `"aaaaaaa".match(/a+?/)`
      - Vs
    - `"aaaaaaa".match(/a+/)`

Subgroups

- ()
  - "Saves" the results of a portion of the overall match
  - Can recall previously matched values with \n
    - Where *n* is a number
  - E.g.,
    - "foofoo".match(/(.*)\1/)
      - Finds a match!
    - "foobar".match(/(.*)\1/)
      - ???
    - "barbaz".match(/(.*)\1/)
      - ???

Handy use of subgroups

- Javascript will allow you to reference matched subgroups in the replace function with $*n*:

```
var re = /(\w+)\s(\w+)/;
var str = 'John Smith';
var newstr = str.replace(re, '$2, $1');
document.write(newstr);
```

Flags

- g
  - Global search
- i
  - Case-insensitive search
- m
  - Multi-line search.
- y
  - Perform a "sticky" search that matches starting at the current position in the target string

Odds and ends

- |
  - Or
  - /red|green/
- (?:x)
  - Matches, but does not save x
- x(?=y)
  - Matches x only if followed by y
- x(?!y)
  - Matches x only if it is not followed by y

Examples

- Write regular expressions to perform the following:
  - Whether a string contains a valid floating point number
  - Whether a string represents a valid date
  - Whether a string represents a valid email address

To wrap up

- When developing a regular expression, consider two different questions:
  - Does it MATCH all of the strings you want it to match?
  - Does it NOT MATCH all of the strings you do not want it to match?
- Mistakes are often made when only one of those questions is considered

Relevant XKCDs

REGEX GOLF:
YOU TRY TO MATCH ONE GROUP BUT NOT THE OTHER.
/M | [TN]|B/ MATCHES STAR WARS SUBTITLES BUT NOT STAR TREK.
COOL.
META-REGEX GOLF:
SO I WROTE A PROGRAM THAT PLAYS REGEX GOLF WITH ARBITRARY LISTS...
UH OH...
META-META-REGEX GOLF:
...BUT I LOST MY CODE, SO I'M GREPPING FOR FILES THAT LOOK LIKE REGEX GOLF SOLVERS.
...AND BEYOND:
REALLY, THIS IS ALL /(META-)*REGEX GOLF/.
NOW YOU HAVE INFINITE PROBLEMS.
NO, I HAD THOSE ALREADY.