Regular Expressions

Class 11

Regular Expressions

- a regular expression (regex or RE) is a coded representation of a pattern of text characters
- a RE is used as the argument of a match
- a text string matches the RE if the RE's pattern exists in the text string

Regular Expressions

- each character in a regular expression is either
  - a metacharacter with a special meaning
  - a "normal" character literally itself

Languages

in the context of web programming REs are used in

- HTML (form elements)
- PHP
- JS

the three are similar, but not identical
we'll start with PHP

RE Delimiters

- every RE must be delimited
- any non-alphanumeric, non-backslash, non-whitespace character is a valid delimiter
- common delimiters are | / # ~ + %
- the delimiter cannot occur in the pattern, so you choose one that you don't need in the pattern

examples:

```
/foobar/
+http://foo.bar+
%[a-zA-Z0-9_-]%
```

Literal Characters

every character is either literally itself or a metacharacter

example literals:

`/foo/` matches `"foobar"` and `"I got fooled"`
`/s t/` matches `"Now is the time"`
`/st/` does not match `"Now is the time"`

Grouping and OR

the vertical bar | is a metacharacter that denotes logical OR

parentheses are metacharacters used for grouping

`/abc|def/` will match any string that contains "abc" and will also match any string that contains "def"

`/(Fred|Wilma) Flintstone/` will match a string that contains "Fred Flintstone" and also a string that contains "Wilma Flintstone"

Wildcard

- the period metacharacter . matches any single character (but not an embedded newline)

`/M. Bell/` matches “Ma Bell” and “Mr Belly” and “ADAM! Bell”

Quantifiers

- the plus sign + metacharacter matches one or more of the thing immediately preceeding it
- the asterisk * matches zero or more of the thing immediately preceeding it
- the question mark ? matches zero or one of the thing immediately preceeding it

`~Go+gle~` matches "Gogle", "Google", "Gooooooogle", etc.
`~Go*gle~` matches "Ggle", "Google", "Gooooooogle", etc.
`~Go?gle~` matches "Ggle" and "Gogle" but not "Google"

Quantifiers

- a number in curly braces {5} matches exactly that number of the thing preceeding it
- two comma-separated numbers in curly braces {3, 5} match exactly that number of the thing preceeding them

`~Go{2, 3}gle~` matches "Google" and "Gooogle" but not "Goooogle"

- {,5} matches up to five things, while {5,} matches five or more things

Character Sets

- character sets allow you to match any of a set of characters
- square brackets [] group characters into a set
  `%[bcd]art%` matches "a barty", "a carty", and "a darty"
- a hyphen matches a range of characters
  `%[a-z]%` matches any lowercase letter
  `%[A-Za-z0-9]%` matches any letter or digit
- many character sets are so common they have special notations
  - \d is any digit, shorthand for [0-9]
  - \w is any of the identifier characters [A-Za-z0-9_]
  - \s is any whitespace character (tab, space, newline)
  - \D is any character that is not a digit
  - \W is the ASCII complement of \w
  - \S is any non-whitespace character

Modifiers

there are many modifiers for regexs

- outside of square brackets:
  - \ escape character
  - ^ assert start of string
  - $ assert end of string
- inside of square brackets:
  - ^ if the first character in a class, it negates the class
    `[^0-9]` matches any character except digits

Global Modifiers

after the trailing delimiter

- `/some pattern/i` matches case insensitive
- `/some pattern/u` pattern and text are treated as utf-8 instead of ASCII

REs in PHP

- we often use the Boolean function `preg_match`
- note the single quotes enclosing the pattern

```
if (preg_match('/^(19|20)\d{2}$/', $year))
{
  ... $year is a valid year in the 20th or 21st century
}
else
{
  ... not a valid year
}
```

RE in PHP

a very common idiom in a PHP program:

```
$year = isset($_POST['year']) &&
        preg_match('/^(19|20)\d{2}$/', $_POST['year']) ?
        $_POST['year'] : '2022';
```

Email Addresses

- it is impossible to regex an email address for valid syntax
- most people use filter_var for this

```
$email = isset($_POST['email']) ?
          filter_var($_POST['email'], FILTER_VALIDATE_EMAIL) :
          '';
```

Regex in HTML Input Elements

- the pattern attribute for input elements supports regex
- works in
  - text
  - search
  - url
  - tel
  - email (but be careful with this)
  - password
- there is no delimiter for HTML REs
- no ^ or $ (both are assumed)

example `http://borax.truman.edu/315/c11/htmlre.php`

Practice

- `http://borax.truman.edu/315/c11/regexpexercises.html`