What is form validation?

- **validation**: ensuring that form’s values are correct
- some types of validation:
  - preventing blank values (email address)
  - ensuring the type of values
    - integer, real number, currency, phone number, Social Security number, postal address, email address, date, credit card number, ...
  - ensuring the format and range of values (ZIP code must be a 5-digit integer)
  - ensuring that values fit together (user types email twice, and the two must match)
A real form that uses validation

Client vs. server-side validation

Validation can be performed:

- **client-side** (before the form is submitted)
  - can lead to a better user experience, but not secure (why not?)
- **server-side** (in PHP code, after the form is submitted)
  - needed for truly secure validation, but slower
- both
  - best mix of convenience and security, but requires most effort to program
An example form to be validated

```html
<form action="http://foo.com/foo.php" method="get">
  <div>
    City:  <input name="city" /> <br />
    State: <input name="state" size="2" maxlength="2" /> <br />
    ZIP:   <input name="zip" size="5" maxlength="5" /> <br />
    <input type="submit" />
  </div>
</form>
```

- Let's validate this form's data on the server...

Basic server-side validation code

```php
$city = $_REQUEST["city"];  
$state = $_REQUEST["state"];  
$zip = $_REQUEST["zip"];  
if (!$city || strlen($state) != 2 || strlen($zip) != 5) {
  ?>
  <h2>Error, invalid city/state submitted.</h2>
  <?php
}
```

- **basic idea:** examine parameter values, and if they are bad, show an error message and abort
- validation code can take a lot of time / lines to write
  - How do you test for integers vs. real numbers vs. strings?
  - How do you test for a valid credit card number?
  - How do you test that a person's name has a middle initial?
  - (How do you test whether a given string matches a particular complex format?)
What is a regular expression?

```
"/^\[a-zA-Z_\-]+@[([a-zA-Z_\-])+\.)+[a-zA-Z]{2,4}$/"
```

- **regular expression** ("regex"): a description of a pattern of text
  - can test whether a string matches the expression's pattern
  - can use a regex to search/replace characters in a string
- regular expressions are extremely powerful but tough to read
  (the above regular expression matches email addresses)
- regular expressions occur in many places:
  - Java: Scanner, String's split method (CSE 143 sentence generator)
  - supported by PHP, JavaScript, and other languages
  - many text editors (TextPad) allow regexes in search/replace

Basic regular expressions

```
"/abc/"
```

- in PHP, regexes are strings that begin and end with /
- the simplest regexes simply match a particular substring
- the above regular expression matches any string containing "abc":
  - YES: "abc", "abcdef", "defabc", ".=abc.=.", ...
  - NO: "fedcba", "ab c", "PHP", ...

Wildcards: .

- A dot . matches any character except a \n line break
  - "/.oo.y/" matches "Doocy", "goofy", "LooNy", ...
- A trailing i at the end of a regex (after the closing /) signifies a case-insensitive match
  - "/mart/i" matches "Marty Stepp", "smart fellow", "WALMART", ...

Special characters: |, (), ^, \

- | means OR
  - "/abc|def|g/" matches "abc", "def", or "g"
  - There's no AND symbol. Why not?
- () are for grouping
  - "/(Homer|Marge) Simpson/" matches "Homer Simpson" or "Marge Simpson"
- ^ matches the beginning of a line; $ the end
  - "/^<!--$/" matches a line that consists entirely of "<!--"
- \ starts an escape sequence
  - many characters must be escaped to match them literally: / \ $ . [ ] ( ) ^ * + ?
  - "/<br \/>/" matches lines containing <br /> tags
Quantifiers: *, +, ?

- * means 0 or more occurrences
  - "/abc*/" matches "ab", "abc", "abcc", "abccc", ...
  - "/a (bc) */" matches "a", "abc", "abcbc", "abcbcbc", ...
  - "/a.*a/" matches "aa", "aba", "a8qa", "a!?_a", ...

- + means 1 or more occurrences
  - "/a (bc) +/" matches "abc", "abcbc", "abcbcbc", ...
  - "/Goo+gle/" matches "Google", "Gooogle", "Goooolge", ...

- ? means 0 or 1 occurrences
  - "/a (bc) ?/" matches "a" or "abc"

More quantifiers: {min, max}

- {min, max} means between min and max occurrences (inclusive)
  - "/a (bc) \{2, 4\} /" matches "abcbc", "abcbcbc", or "abcbcbcbc"

- min or max may be omitted to specify any number
  - {2,} means 2 or more
  - {, 6} means up to 6
  - {3} means exactly 3
Character sets: [ ]

- [ ] group characters into a character set; will match any single character from the set
  - "/[bcd]art/" matches strings containing "bart", "cart", and "dart"
  - equivalent to "/(b|c|d)art/" but shorter
- inside [], many of the modifier keys act as normal characters
  - "/[what[!*?]*/" matches "what", "what!", "what??!", "what??!", ...
- What regular expression matches DNA (strings of A, C, G, or T)?
  - "/[ACGT]+/"

Character ranges: [start-end]

- inside a character set, specify a range of characters with -
  - "/[a-z]/" matches any lowercase letter
  - "/[a-zA-Z0-9]/" matches any lower- or uppercase letter or digit
- an initial ^ inside a character set negates it
  - "/[^abcd]/" matches any character other than a, b, c, or d
- inside a character set, - must be escaped to be matched
  - "/[+\-]?[0-9]+/" matches an optional + or -, followed by at least one digit
- What regular expression matches letter grades such as A, B+, or D-?
  - "/[ABCDF][+\-]*/"
Escape sequences

- special escape sequence character sets:
  - \d matches any digit (same as \[0-9\]); \D any non-digit (\[^0-9\])
  - \w matches any “word character” (same as \[a-zA-Z_0-9\]); \W any non-word char
  - \s matches any whitespace character (, \t, \n, etc.); \S any non-whitespace
- What regular expression matches dollar amounts of at least $100.00 ?
  - "/\$\d{3,}\d{2}/"

Regular expressions in PHP (PDF)

- regex syntax: strings that begin and end with /, such as "\/[AEIOU]+/"

<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>preg_match(regex, string)</td>
<td>returns TRUE if string matches regex</td>
</tr>
<tr>
<td>preg_replace(regex, replacement, string)</td>
<td>returns a new string with all substrings that match regex replaced by replacement</td>
</tr>
<tr>
<td>preg_split(regex, string)</td>
<td>returns an array of strings from given string broken apart using the given regex as the delimiter (similar to explode but more powerful)</td>
</tr>
</tbody>
</table>
Regular expression example

```php
# replace vowels with stars
$str = "the quick brown fox";

$str = preg_replace("/[aeiou]/", "*", $str);
    # "th* q**ck br*wn f*x"

# break apart into words
$words = preg_split("/[ ]+/", $str);
    # ("th*", "q**ck", "br*wn", "f*x")

# capitalize words that had 2+ consecutive vowels
for ($i = 0; $i < count($words); $i++) {
    if (preg_match("/\*{2,}/", $words[$i])) {
        $words[$i] = strtoupper($words[$i]);
    }
}
    # ("th*", "Q**CK", "br*wn", "f*x")

- notice how \ must be escaped to \\n
PHP form validation w/ regexes

```php
$state = $_REQUEST["state"];
if (!preg_match("/[A-Z]{2}/", $state)) {
    ?>
    <h2>Error, invalid state submitted.</h2>
    <?php
}

- using preg_match and well-chosen regexes allows you to quickly validate query parameters against complex patterns