Form Validation

What is form validation?

- **validation:** ensuring that form's values are correct
- some purposes 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

oogle Calendar | How the Java Virtual Ma... | http://localh.../testfile.php | Web Progra

htwatchers.com/SignupVersions/registration/StepOne.aspx

WeightWatchers

Important message: Please review and correct the information below.

Create Your Registered User Account Login

First name: Xenia

Last name:

Please enter your last name. It's required information.

Your birthdate: Month Day Year

Please select your month, day, and year of birth. It's required information.

Your gender: Female Male

Please enter your gender. It's required information.

State: (Select One)

Please choose a state. It's required information.

Zip code

Please enter a 5-digit ZIP code. It's required information.

E-mail:

Please enter your email address in the following format: abc@example.com. It's required information.

Re-enter e-mail:

vious Highlight all Match case

Client vs. server-side validation

- Validation places:
  - **client-side** (before the form is submitted)
    - can lead to a better user experience, but not secure as client can change the front-end's code.
    - Can be achieved by attribute (e.g., required, maxlength) or by JS code (inputElement.value).
  - **server-side** (in PHP code, after the form is submitted)
    - needed for secured validation as client do not have access to the server side code
    - Slower as need to hit the server every time.

Front-end form validation example


```
<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>
```

HTML

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

Basic server-side validation code


```
$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
}
?>
```

PHP

- basic idea: examine parameter values, and if they are bad, show an error message and abort

Basic server-side validation code

- 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?
  - **Solution is regular expression**

Regular expression

Regular expressions

- A regular expression is a sequence of characters that forms a search pattern. In PHP, it's used for pattern matching and string manipulation.
- For example, determining whether an input field contains an email or not.
- PHP has two main sets of functions for working with regular expressions:
  - POSIX (deprecated): example, ereg(), eregi(), etc.
  - **PCRE (Perl-Compatible Regular Expressions)**

Basic Regular Expression

"/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", ...
- A trailing i at the end of a regex (after the closing /) signifies a case-insensitive match
  - Example: "/xen/i" matches “Xenia", “xenophobic", “Xena the warrior princess", “XEN technologies” ...

Some Perl functions

- preg_match() # Checks if a pattern matches
- preg_match_all() # Finds all matches (return count)
- preg_replace() # Replaces matches (return string)

- Example

```
$pattern = "/php/i";  # 'i' means case-insensitive
                      # modifier and  '/' is delimiter
$text = "I love PHP";
if (preg_match($pattern, $text)) {
    echo "Match found!";
}
```

Common Regex Elements

Symbol	Meaning
.	Any character except newline
^	Start of string, (or as not)
$	End of string
\d	Digit (0-9)
\w	Word character (a-z, A-Z, 0-9, _)
\s	Whitespace
+	One or more
*	Zero or more
?	Zero or one
[]	Match any one inside
()	Group
{n,m}	Between n and m times

Use of '[]'

- Finds the characters inside the brackets

- Example

```
<?php
    $txt = "Hello world";
    $pattern = "/[dl]/";
    echo preg_match_all($pattern, $txt);
?>
<p>The matches were found here:</p>
<?php
echo preg_replace($pattern, "#", $txt);
?>
```

Output:

- **4**
- **He##o wor##**

Q: What if $pattern = "/[d-l]/";

- **5**
- **H###o wor##**

Q: What if $pattern = "/[^d-l]/";

- **6**
- **#ell#####ld**

More examples


```
[a-z]at                 #cat, rat, bat…
[aeiou]                 #between a,e,i,o,u
[a-zA-Z]                #between a to z or A to Z
[^a-z]                  #not a-z
(very )*large           #large, very very very large…
(very )+large           #very large, very very large…
(very){1, 3}            #counting "very" up to 3
^bob                    #bob at the beginning
com$                    #com at the end
```

PHPRegExp

Note: Put all the patterns inside '/' and '/'

Special characters: ., |, (), ^, \

- A dot . matches any character except a \n line break
  - "/.oo.y/" matches "Doocy", "goofy", "LooNy", ...
- | 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 "<!--"

Special characters: |, (), ^, \

- \ 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", "Goooogle", ...
- ? 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)?

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

Character ranges: [start-end]

- 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- ?

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 ?

Regular expressions example


```
echo preg_match ('/test/', "a test of preg_match");
echo preg_match ('/tutorial/', "a test of preg_match
");

$matchesarray[0] = "http://www.tipsntutorials.com/"
$matchesarray[1] = "http://"
$matchesarray[2] = "www.tipsntutorials.com/"
preg_match ('/(http://)(.*)/', "http://www.tipsntuto
rials.com/", $matchesarray)
```

PHP

Regular expressions example


```
# 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")
```

PHP

PHP form validation w/ regexes


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

PHP

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

Task

- Write a PHP script that tests whether an e-mail address is input correctly. Test it using valid and invalid addresses
- Write a regular expression validate number between 1971 to 2025 and explain the importance of use ^ and $ at the same time.

Task

You are given a text containing employee codes, where each code follows the format EMP-XXX-YYYY, with EMP- as a fixed prefix, XXX being a 3-letter department code (HRD, ENG, or MKT), and YYYY being a 4-digit employee number starting with 1, 2, or 3. Write a PHP regular expression to validate and extract all correct employee codes from a block of text. For example, from the input "New employees are EMP-HRD-1456, EMP-ENG-3567, EMP-MKT-4890, EMP-IT-1234.", the valid extracted employee codes should be EMP-HRD-1456, EMP-ENG-3567, and EMP-MKT-4890.

```
// ------------- CODE ------------------
<?php
function
extractValidEmployeeCodes($text) {
$pattern = ‘put expression here';
preg_match_all($pattern, $text,
$matches);
return $matches[0];
}
$text = "New employees are EMP-HRD-
1456, EMP-ENG-3567, EMP-MKT-4890,
EMP-IT-1234.";
$validEmployeeCodes =
extractValidEmployeeCodes($text);
print_r($validEmployeeCodes);
?>
```