CPET 499/ITC 250 Web Systems

Chapter 15
Error Handling and Validation

Text Book:
*** Fundamentals of Web Development, 2nd, by Randy Connolly and Ricardo Hoar, published by Pearson**

Purdue University For Wayne
School of Polytechnic

Paul I-Hai Lin, Professor
http://www.etcs.pfw.edu/~lin

Topics

- Why the different types of errors and how they differ from exceptions
- The different forms of error reporting in PHP
- How to handle errors and exceptions
- What regular expressions are and how to use them in JavaScript and PHP
- Some best practices in design user input validation
- How to validate inputs in HTML5, JavaScript, and PHP

Errors and Exceptions

- **Exceptions**
  - For run time errors handling
  - Object-oriented construct:
    - try … catch … exceptions
- **Errors**
  - Expected errors
  - Warnings
  - Fatal Errors

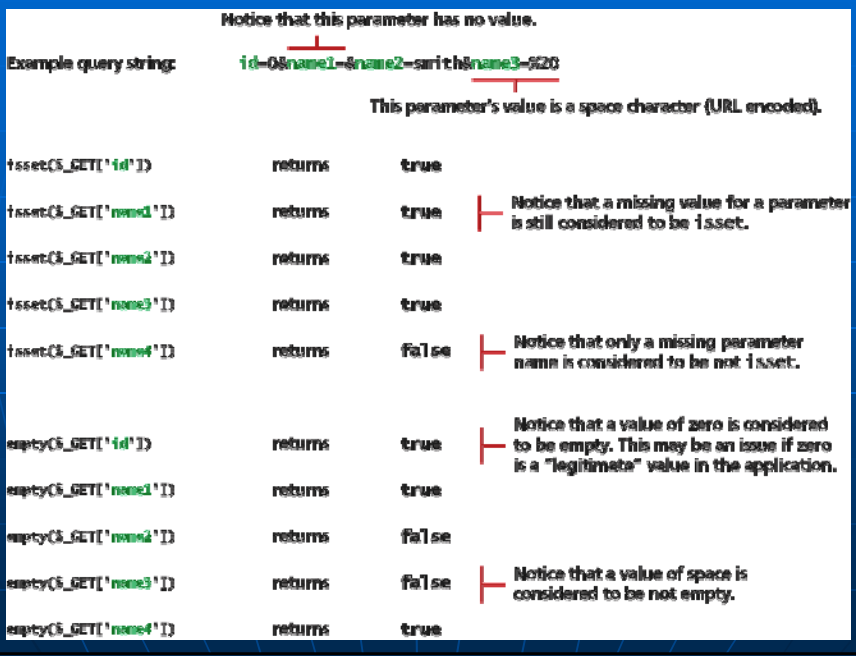
Errors

- **Types of Errors**
  - **Expected errors**
    - **An error that routinely occurs during running an application.**
    - **User data entering errors**
    - **isset() can be used for testing the value of a variable**
    - **empty() for checking query string values:**
      - **Return "TRUE" if a variable is NULL, FALSE, ZERO, or an EMPTY STRING**
    - **is_numeric() for testing a query string parameter to be numeric**
  - **Warnings**
  - **Fatal errors**

Errors

- Types of Errors
  - Expected errors
  - Warnings
    - PHP warning messages may or may not be displayed
    - It will halt the execution of the page
  - Fatal errors
    - Serious errors that the execution of the program will terminate unless handled in other proper ways

Figure 15.1 Comparing isset() and empty() with query string parameters



Listing 15.1 Testing a query string: existence & numeric

```
<?php
//Listing 15.1 Testing a query string to see if it exists and is numeric
$id = $_GET['id'];
if (!empty($id) && is_numeric($id) )
{
// use the query string since it exists and is a numeric value
//...}
?>
```


PHP Error Reporting

- **Runtime Configuration**
  - http://php.net/manual/en/errorfunc.configuration.php
- **Three main error reporting flags**
  - error_reporting
    - error_reporting(E_ALL)
    - errort_reporting = E_ALL … in php.ini File
  - display_errors
    - Ini_set('display_errors', '0');
  - log_errors

PHP Error Reporting

error_reporting specifies which type of errors are to be reported:

```
ini_set('log_errors','1');
```

It can also be set within the php.ini file:

```
log_errors = On
```


The Display Error Setting

The display_error setting specifies whether error messages should or should not be displayed in the browser:

```
ini_set('display_errors','0');
```

It can also be set within the php.ini file:

```
display_errors = Off
```

PHP Error and Exception Handling (Procedural)

```
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME);
$error = mysqli_connect_error();
if ($error != null) {
        // handle the error
        …
}
```


PHP Error and Exception Handling (Objected-Oriented)

```
Exception throwing function (for illustration purposes)
function throwException($message = null,$code = null) {
        throw new Exception($message,$code);
}
try {
// PHP code here
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME)
        or throwException("error"); …
}
catch (Exception $e) {
        echo ' Caught exception: ' . $e->getMessage();
        echo ' On Line : ' . $e->getLine();
        echo ' Stack Trace: '; print_r($e->getTrace());
} finally {
        // PHP code here that will be executed after try or after catch
}
```

Regular Expressions

- **Regular Expressions - Definition**
  - A regular expressions is a set of special characters that define a pattern to be matched in strings.
  - Found in Unix Utilities: vi, grep, sed
  - In Web applications, we want to match
    - **A phone number**
    - **Zip code**
    - **Email address, etc**

Regular Expressions

- **Perl Regular Expression**, https://perldoc.perl.org/perlre.html
- **JavaScript Regular Expression**
  - Special characters, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions
  - RegExp(), https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp
- **PHP Regular Expression,** http://php.net/manual/en/reference.pcre.pattern.syntax.php
- **MySQL Regular Expression**:
  - Pattern Matching, https://dev.mysql.com/doc/refman/5.7/en/pattern-matching.html
  - Regular Expressions, https://dev.mysql.com/doc/refman/5.7/en/regexp.html
-

Regular Expressions

- **PHP Regular Expressions,** http://php.net/manual/en/reference.pcre.pattern.syntax.php
- **MySQL Regular Expressions**:
  - Pattern Matching, https://dev.mysql.com/doc/refman/5.7/en/pattern-matching.html
  - Regular Expressions, https://dev.mysql.com/doc/refman/5.7/en/regexp.html
- **.NET Regular Expression,** https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expressions

Regular Expression Syntax

- A regular expression consists of two types of characters
  - Literals – characters to be matched in the target string
  - Metacharacters – special symbols that acts as a command to the regular expression parser
- 14 Metacharacters (characters with special meaning)

.	[	]	\	(	)	^
$	|	*	?	{	}	+

Regular Expression Examples

- PHP Example: begin and end with backward slahes
  - $pattern = ‘/ran/;
  - ‘randy Connolly’; ‘Sue ran to store’; “I would like a cranberry’

```
$check = ‘Sue ran to the store’;
if (preg_match($pattern, $check)){
  echo ‘Match found!’;
```

- JavaScript Example: begin and end with forward slashes

```
var pattern = /ran/;
If (pattern.test(‘Sue ran to the store’)) {
Document.write(‘Match found’);}
```


Common Regular Expression Patterns

Pattern	Description
^ qwerty $	Match the entire string between the ^ and the $ symbols; between the very start and end of.
\t	Matches a tab character
\n	Matches a new-line character
.	Matches any character other than \n
[qwerty]	Matches any single character of the set contained within the brackets
[^qwerty]	Matches any single character not contained within the brackets

Common Regular Expression Patterns

Pattern	Description
[a-z]	Matches any single character within the range of characters
\w	Matches any word character, equivalent to {a-zA-Z0-9}
\W	Matches any nonword character
\s	Matches any white-space character
\S	Matches any nonwhite-space character

Common Regular Expression Patterns

Pattern	Description
\d	Matches any digit
\D	Matched any non-digit
*	Indicates zero or more matches
+	Indicates one or more matches
?	Indicates zero or one match
{n}	Indicates exactly n matches
{n,}	Indicates n or more matches
{n.m}	Indicates at least n but no more than m matches
|	Matches any one of the terms separated by the | character. Equivalent to Boolean OR
( )	Groups a subexpression. Grouping can make a regular expression easier to understand

Extended Regular Expression Examples

- **^\d{3}-\d{4}$** : Matches any string containing three numbers, follow by a dash, followed by four numbers without any other characters
- **^[2-9]\d{2}-\d{4}$** : Matched a phone number that would NOT allow the first digit in the phone number to be a zero (“0”) or a one (“1”).
- **^[2-9]\d{2}[-\s\.]\d{4}$** :Allows a single space for example (481 6339), a period (481,6339), or a dash (481-6339) between the two sets of numbers.
- **^[2-9]\d{2}[-\s\.]\s*\d{4}$** :Allow multiple spaces (but only one single dash or period) in our phone.

Listing 15.7 A phone number validation without regular expression (partial listing)

```
//Listing 15.7 A phone number validation script without regular
//expressions
var phone=document.getElementById("phone").value;
var parts = phone.split("."); // split on .
if (parts.length !=3)  {
  parts = phone.split("-"); // split on –
}
```

Listing 15.7 A phone number validation without regular expression (partial listing)

```
if (parts.length == 3) {
  var valid=true; // use a flag to track validity
  for (var i=0; i < parts.length; i++) {
   // check that each component is a number
   if (!isNumeric(parts[i])) {
   alert( "you have a non-numeric component");
   valid=false;    }
    else
     { // depending on which component make sure it's in range
       if (i<2) {
      if (parts[i]<100 || parts[i]>999) {
        valid=false; }  }
    else { if (parts[i]<1000 || parts[i]>9999) {  valid=false;      }    }
   }  // end if isNumeric
  } // end for loop
```


Listing 15.7 A phone number validation without regular expression (partial listing)

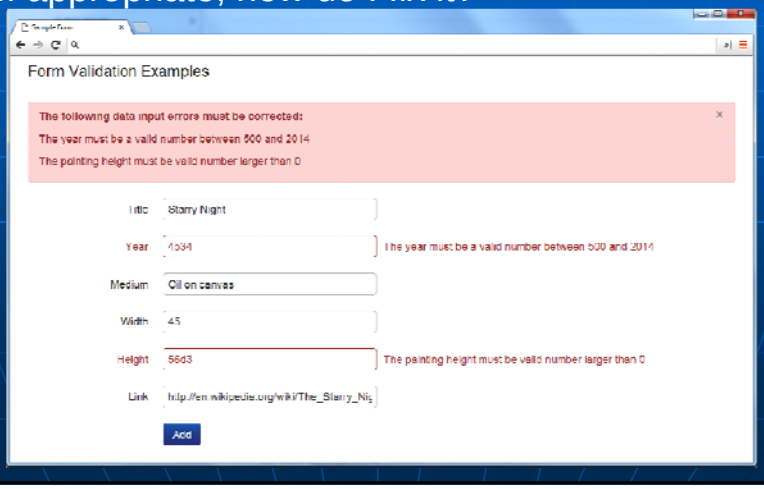
```
if (valid) {
    alert(phone + "is a valid phone number");
   }
 }alert ("not a valid phone number");
```

Validating User Input

- Types of Input Validation
  - **Required Information**
    - Data fields just cannot be left empty
    - Emails, phones, user name, passwords, etc.
  - **Correct Data Type**
    - Numeric, Dates, Strings, etc.
  - **Correct Format**
    - Postal codes, Credit Card numbers, Social security numbers, etc.
  - **Comparison:** passwords, entered values
  - **Range Check**: Dates, number
  - **Custom Validation:**

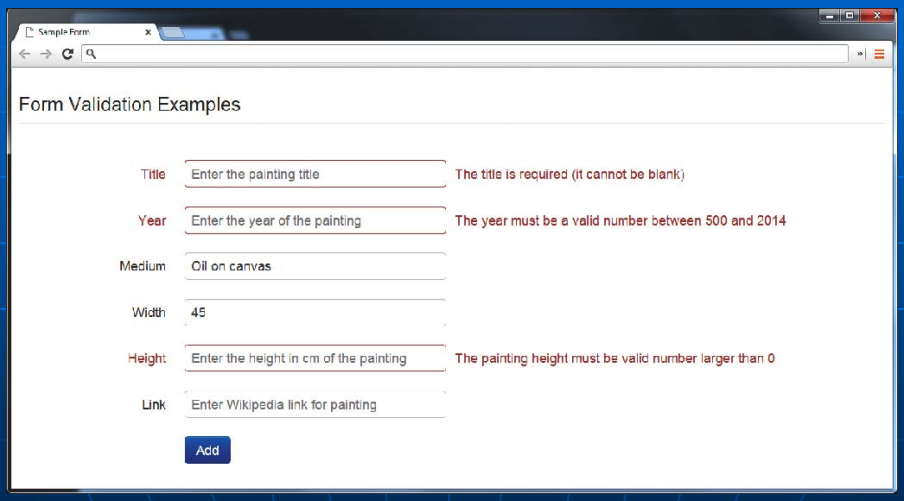
Fig. 15.2 Displaying error messages

- What is the Problem?
- Where is the Problem?
- If appropriate, how do I fix it?



Notifying the User

- Fig. 15.3 Indicating where an error is located



How to Reduce Validation Errors

- Common ways that minimize user validation errors
  - Using **pop-up JavaScript Alert** Messages (or other popup),
  - Provide **textual hints** to the user on the form itself, Fig. 15-4
  - Using **tool tips or pop-over** to display context-sensitive help, Fig. 15. 5
  - Providing **JavaScript-based input mask**, Fig. 15.6

Fig. 15.4 Providing textual hints

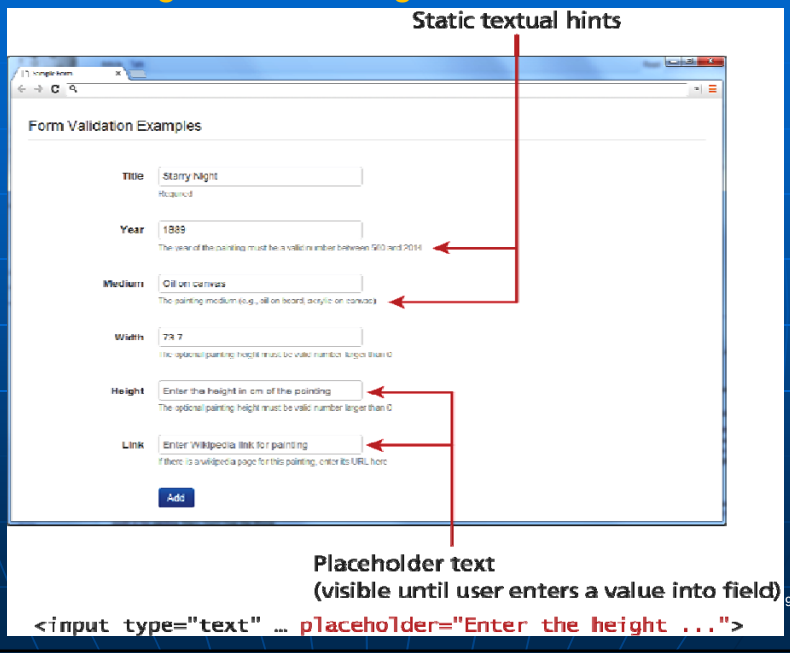


Fig. 15.5 Using tool tips or pop-over

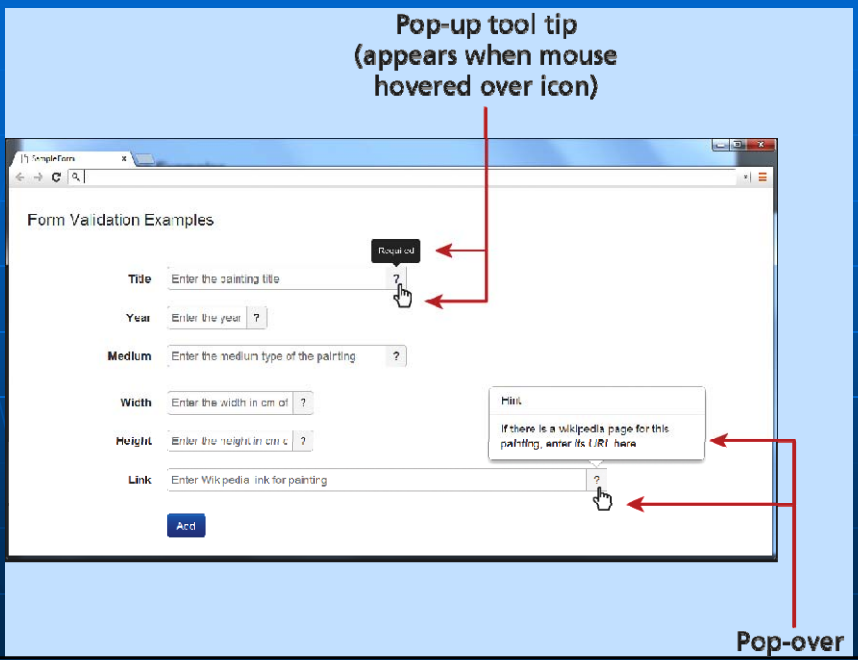
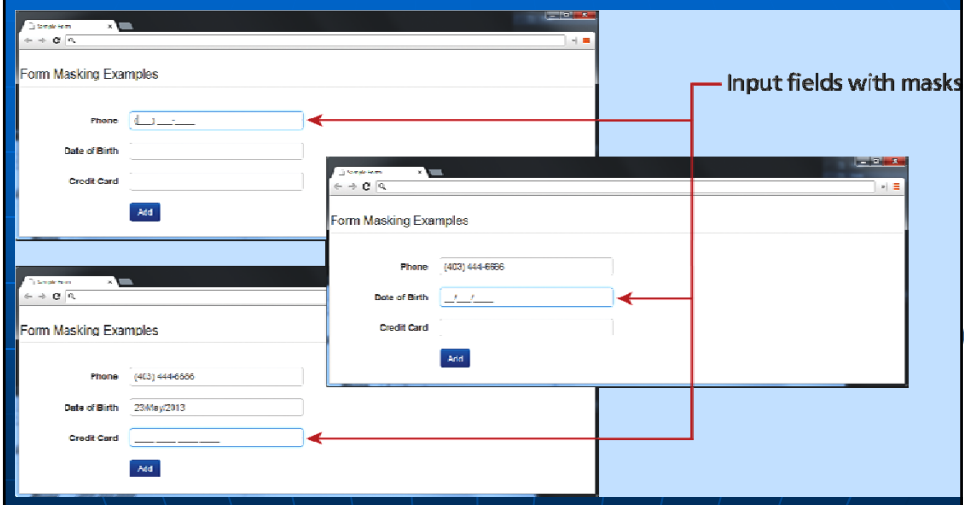


Fig. 15.6 Providing JavaScript-based input mask



Where to Perform Validation

- Performing basic validation with client browsers with HTML 5
- Using JavaScript on the client side (may be turned off on the user's browser)
- Server side validation (always)
  - For example: Validation at the PHP Level

Fig. 15.7 Visualizing levels of validation

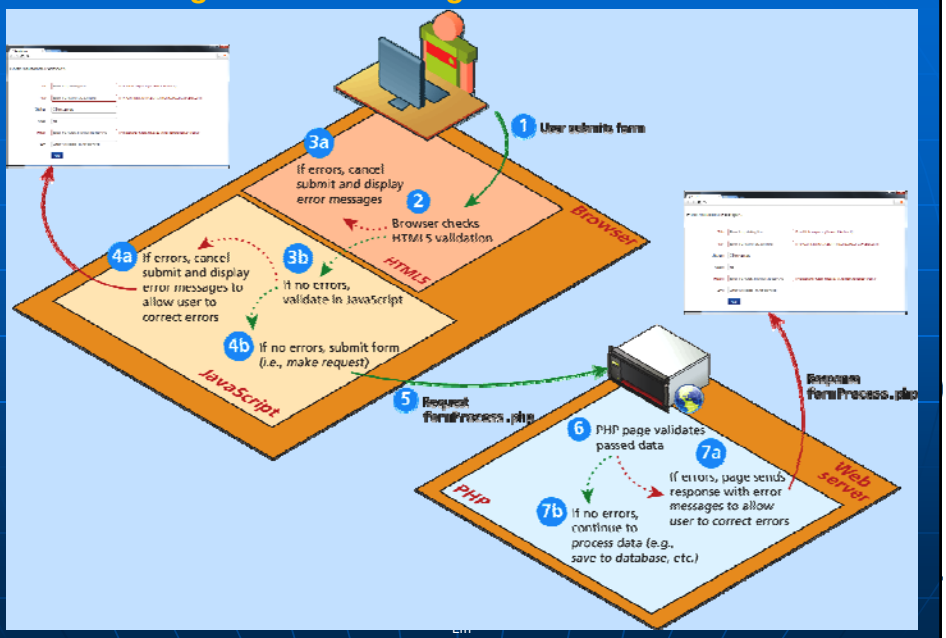


Fig. 15.8 Example form to be validated

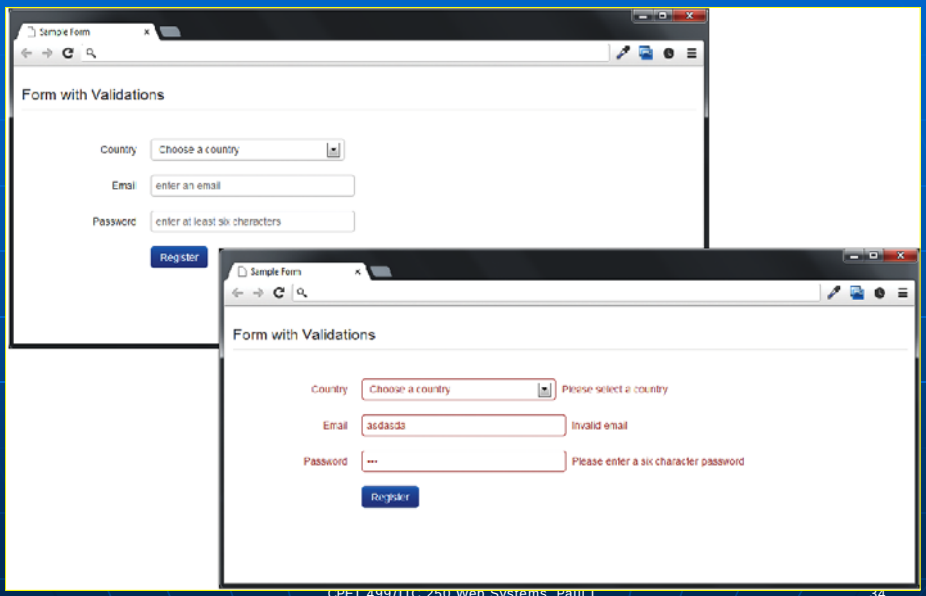
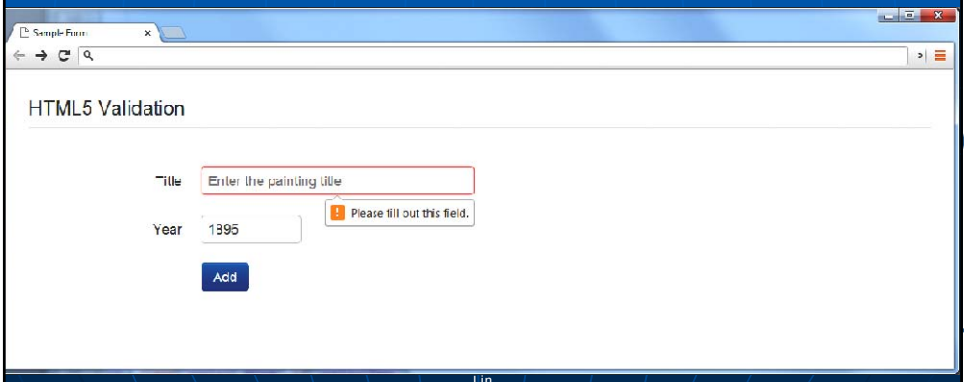


Fig. 15.9 HTML5 browser validation

- Add “required attribute” to input element
- <form id=“sampleForm” method=“…” action=“…” nonvalidate>
- http://www.w3.org/TR/html5/forms.html



Lin

Listing 15.8 Example form (validationform.php) to be validated

```
<?php
//Listing 15.8 Example form (validationform.php) to be validated?>
<form method="POST"  action=“validationform.php class="form-horizontal" id="sampleForm" >
<fieldset><legend>Form with Validations</legend>
<div class="control-group" id="controlCountry">
<label class="control-label" for="country">Country</label>
  <div class="controls"><select id="country" name="country" class="input-xlarge"><option value="0">Choose a country</option><option value="1">Canada</option><option value="2">France</option><option value="3">Germany</option><option value="4">United States</option></select>
<span class="help-inline" id="errorCountry"></span>
</div>
</div>
```

Listing 15.8 Example form (validationform.php) to be validated

```
<div class="control-group" id="controlEmail">
 <label class="control-label" for="email">Email</label>
  <div class="controls">
  <input id="email" name="email" type="text"
      placeholder="enter an email" class="input-xlarge“  required>
  <span class="help-inline" id="errorEmail"></span>
</div>
```


Listing 15.8 Example form (validationform.php) to be validated

```
<div class="control-group" id="controlPassword">
 <label class="control-label" for="password">Password</label>
  <div class="controls">
 <input id="password" name="password" type="password"
  placeholder="enter at least six characters" class="input-xlarge"
  required>
 <span class="help-inline" d="errorPassword"></span>
  </div>
</div>
```

Listing 15.8 Example form (validationform.php) to be validated

```
<div class="control-group">
<label class="control-label" for="singlebutton"></label>
 <div class="controls"><button id="singlebutton"
  name="singlebutton" class="btn btn-primary">
 Register</button>
 </div>
</div>
</fieldset>
</form>
```


Validation at the JavaScript Level

```
<div class="control-group">
<label class="control-label"
for="singlebutton"></label>
<div class="controls">
<button id="singlebutton"
name="singlebutton" class="btn btn-
primary">Register</button>
</div></div></fieldset></form>
```

Validation at the JavaScript Level

- Initialize the validation function once an element loses its focus

```
function init(){
 var sampleForm=document.getElementById('sampleForm');
 sampleForm.onsubmit = validateForm;
 }
// Call the init() function once all the html has been loaded
window.onload = init;
```

- Password input element is between 8 and 16 character

```
var passReg = /^[a-zA-Z]\w{8,16}$/;
if(! passReg.test(password.value){
 // provide some type of error message
```

- Listing 15.9 Complete JavaScript Validation

Validation at the PHP Level

- Listing 15.10 ValidationResult Class
- Listing 15.11 PHP form validation
- Listing 15.12 Revised form with PHP validation messages

Summary and Conclusion

Q/A ?