Error Handling and Validation

Chapter 15

Randy Connolly and Ricardo Hoar

Chapter 15

1. What Are Errors and Exceptions?
2. PHP Error Reporting
3. PHP Error and Exception Handling
4. Regular Expressions
5. Validating User Input
6. Where to Perform Validation
7. Summary

Chapter 15

1. What Are Errors and Exceptions?
2. PHP Error Reporting
3. PHP Error and Exception Handling
4. Regular Expressions
5. Validating User Input
6. Where to Perform Validation
7. Summary

What Are Errors and Exceptions?

Types of Errors

- Expected errors
- Warnings
- Fatal errors

What Are Errors and Exceptions?

Types of Errors

Notice that this parameter has no value.

Example query string: `id=0&name1=&name2=smith&name3=%20`

This parameter's value is a space character (URL encoded).

`isset($_GET['id'])`	returns	`true`	
`isset($_GET['name1'])`	returns	`true`	Notice that a missing value for a parameter is still considered to be `isset`.
`isset($_GET['name2'])`	returns	`true`	
`isset($_GET['name3'])`	returns	`true`	
`isset($_GET['name4'])`	returns	`false`	Notice that only a missing parameter name is considered to be not `isset`.
`empty($_GET['id'])`	returns	`true`	Notice that a value of zero is considered to be empty. This may be an issue if zero is a "legitimate" value in the application.
`empty($_GET['name1'])`	returns	`true`	
`empty($_GET['name2'])`	returns	`false`	
`empty($_GET['name3'])`	returns	`false`	Notice that a value of space is considered to be **not** empty.
`empty($_GET['name4'])`	returns	`true`	

What Are Errors and Exceptions?

Exceptions

An **exception** refers to objects that are of type Exception and which are used in conjunction with the object-oriented try . . . catch language construct for dealing with runtime errors.

Chapter 15

1. What Are Errors and Exceptions?
2. PHP Error Reporting
3. PHP Error and Exception Handling
4. Regular Expressions
5. Validating User Input
6. Where to Perform Validation
7. Summary

PHP Error Reporting

The error_reporting Setting

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

PHP Error Reporting

The display_errors 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 Reporting

The log_errors Setting

The log_errors setting specifies whether error messages should or should not be sent to the server error log.

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

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

```
log_errors = On
```

Chapter 15

PHP Error and Exception Handling

Procedural Error Handling

```
$connection = mysqli_connect(DBHOST, DBUSER,
DBPASS, DBNAME);

$error = mysqli_connect_error();

If ($error != null)  {

        // handle the error

        ...

}
```

PHP Error and Exception Handling

Object-Oriented Exception Handling

```
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
}
```

Chapter 15

Regular Expressions

- A regular expression is a set of special characters that define a pattern.
- Regular expressions are a concise way to eliminate the conditional logic that would be necessary to ensure that input data follows a specific format.
- PHP, JavaScript, Java, the .NET environment, and most other modern languages support regular expressions (each slightly different)

Regular Expressions

Regular Expression Syntax

- A **literal** is just a character you wish to match in the target
- A **metacharacter** is a special symbol that acts as a command to the regular expression parser
  - . [ ] \ ( ) ^ $ | * ? { } +
- Regular Expression Patterns can be combined to form complex expressions

Regular Expressions

Regular Expression Syntax Patterns

- **^ qwerty $** If used at the very start and end of the regular expression, it means that the entire string (and not just a substring) must match the rest of the regular expression contained between the ^ and the $ symbols.
- **\t** Matches a tab character.
- **\n** Matches a new-line character.
- **.** Matches any character other than \n.

Regular Expressions

Regular Expression Syntax Patterns

- **[qwerty]** Matches any single character of the set contained within the brackets.
- **[^qwerty]** Matches any single character not contained within the brackets.
- **[a-z]** Matches any single character within range of characters.
- **\w** Matches any word character. Equivalent to [a-zA-Z0-9].
- **\W** Matches any nonword character.

Regular Expressions

Regular Expression Syntax Patterns

- **\s** Matches any white-space character.
- **\S** Matches any nonwhite-space character.
- **\d** Matches any digit.
- **\D** Matches any nondigit.
- * Indicates zero or more matches.
- + Indicates one or more matches.
- ? Indicates zero or one match.

Regular Expressions

Regular Expression Syntax Patterns

- **{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.

Regular Expressions

Extended Example

```
^(\(?\s*\d{3}\s*[\)–\.]?\s*)?[2-9]\d{2}\s*[–\.]\s*\d{4}$
```

Chapter 15

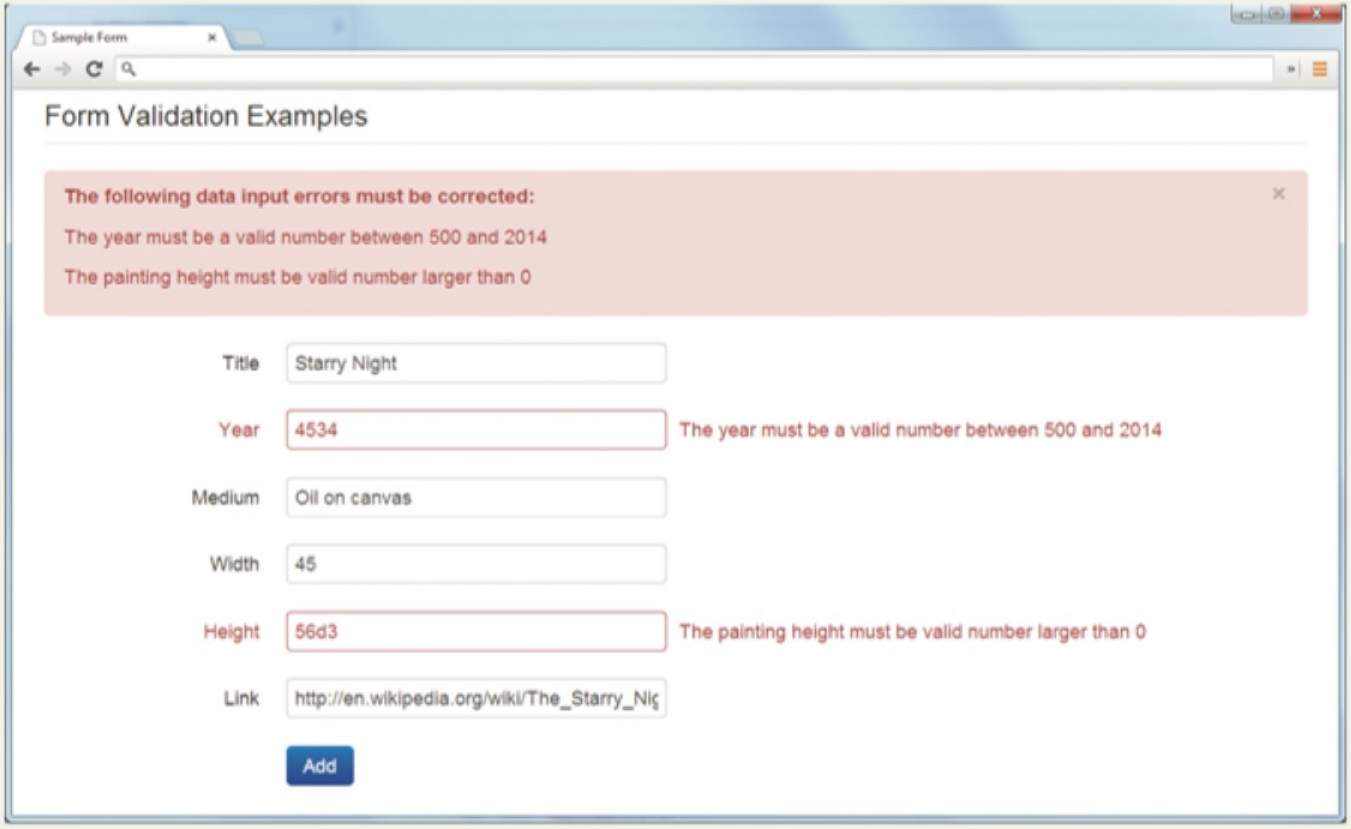
Validating User Input

Types of Input Validation

- Required information
- Correct data type
- Correct format
- Comparison
- Range Check
- Custom

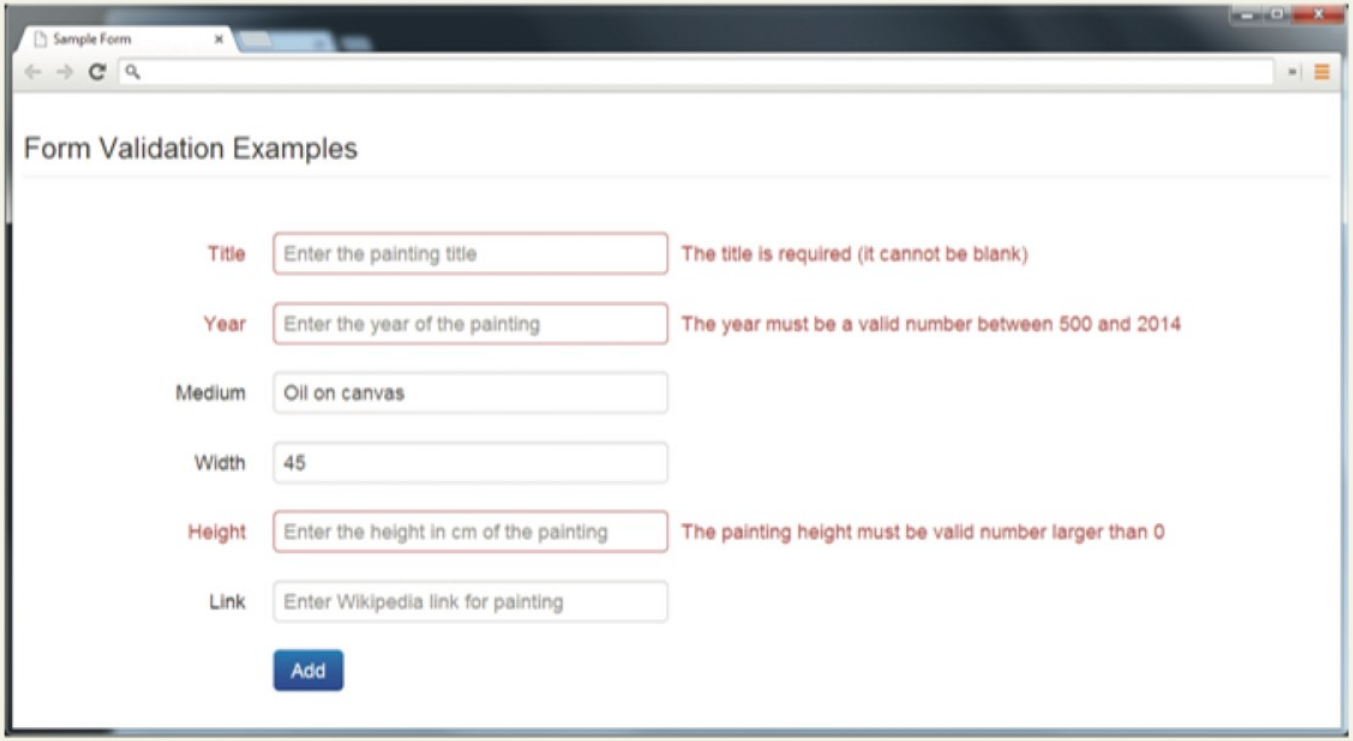
Validating User Input

Notifying the User



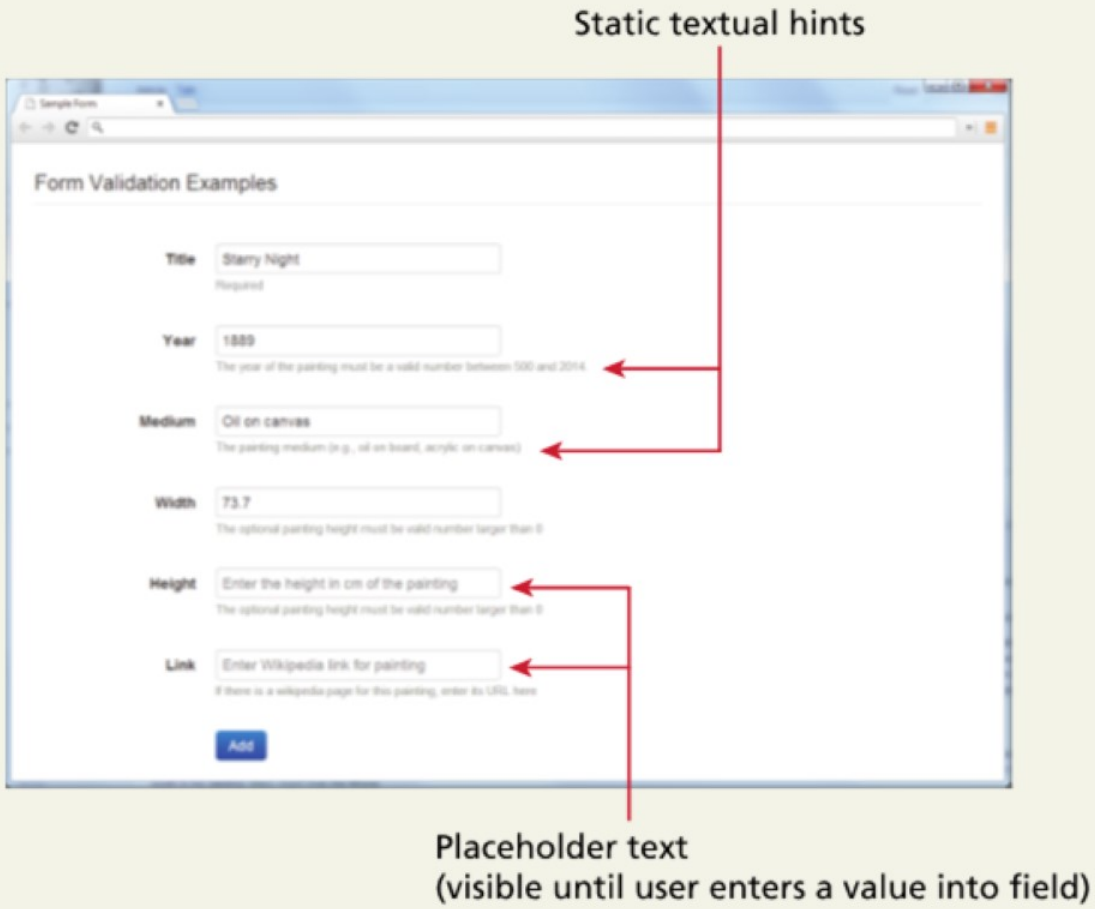
Validating User Input

How to Reduce Validation Errors – show where error located



Validating User Input

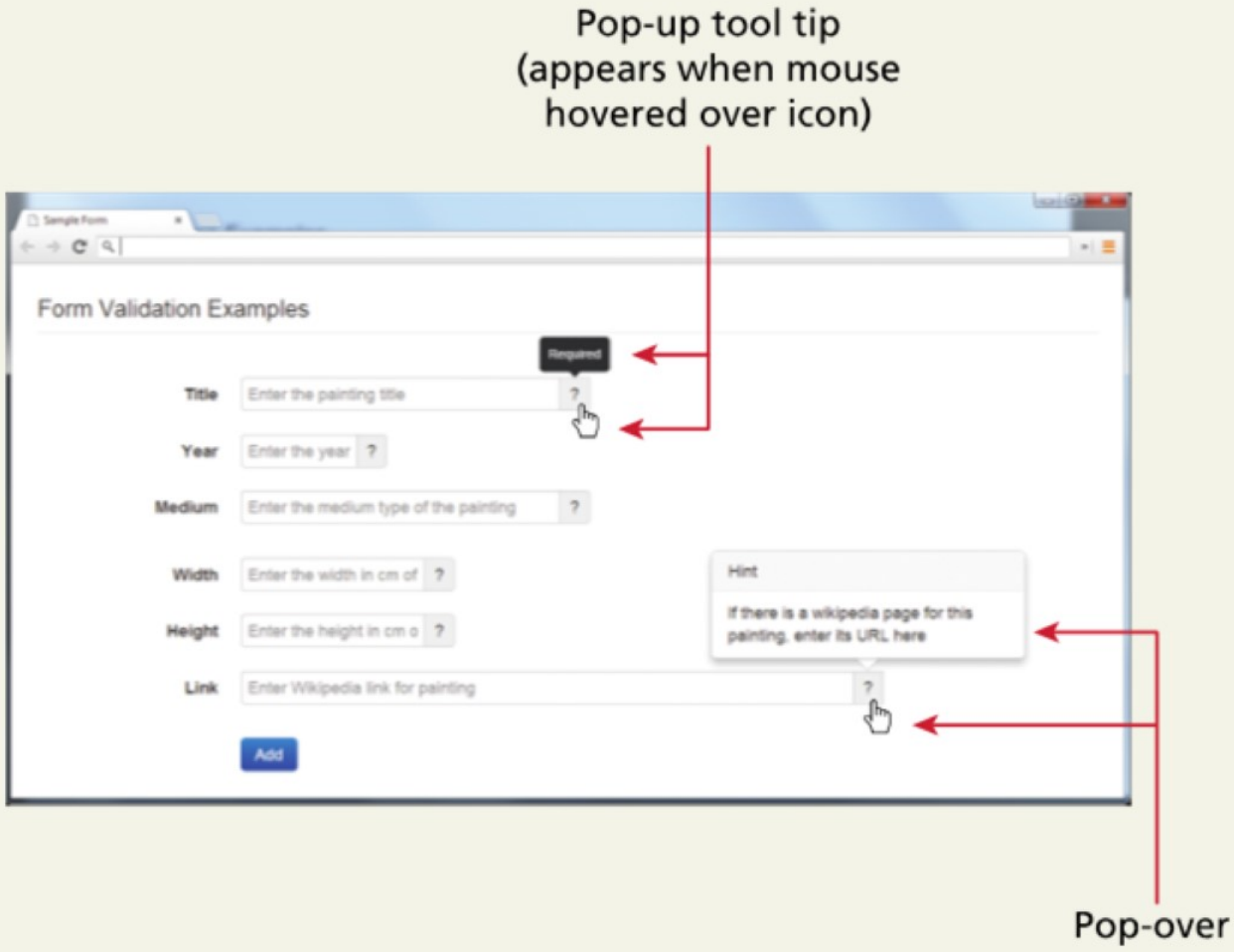
How to Reduce Validation Errors – providing textual hints




```
<input type="text" … placeholder="Enter the height ...">
```

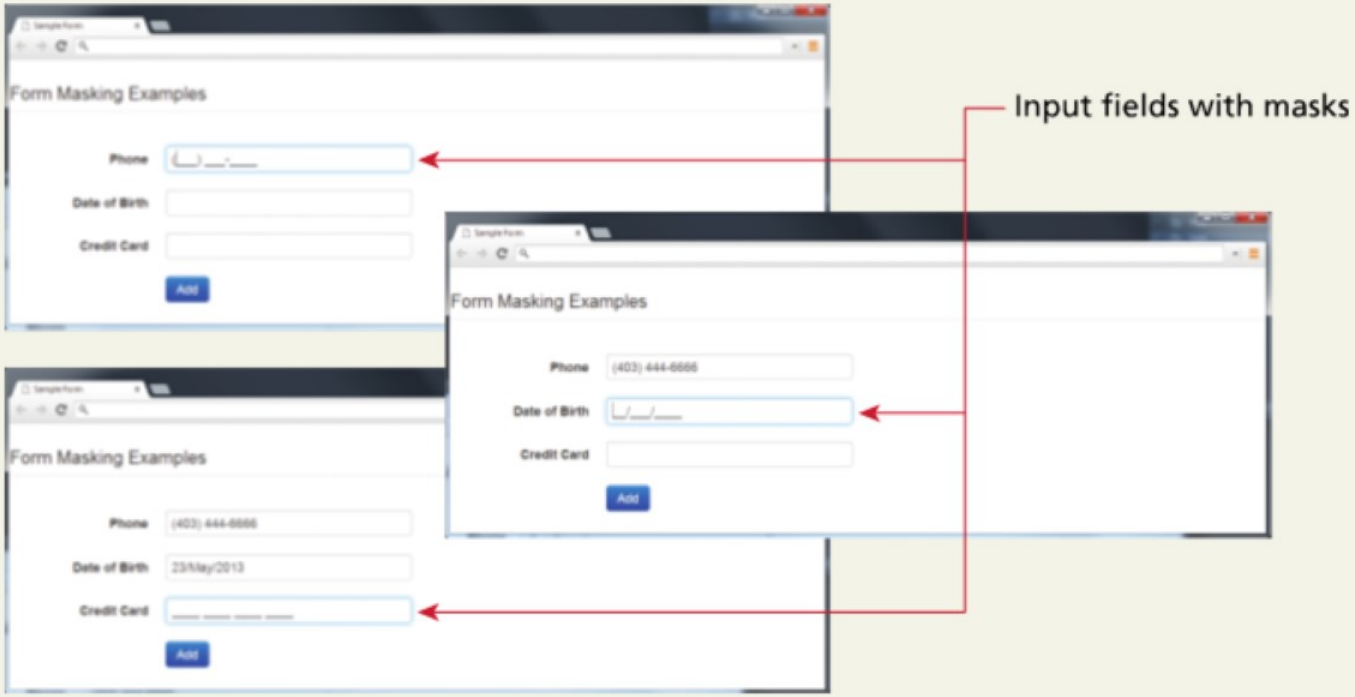
Validating User Input

How to Reduce Validation Errors – use tool tips



Validating User Input

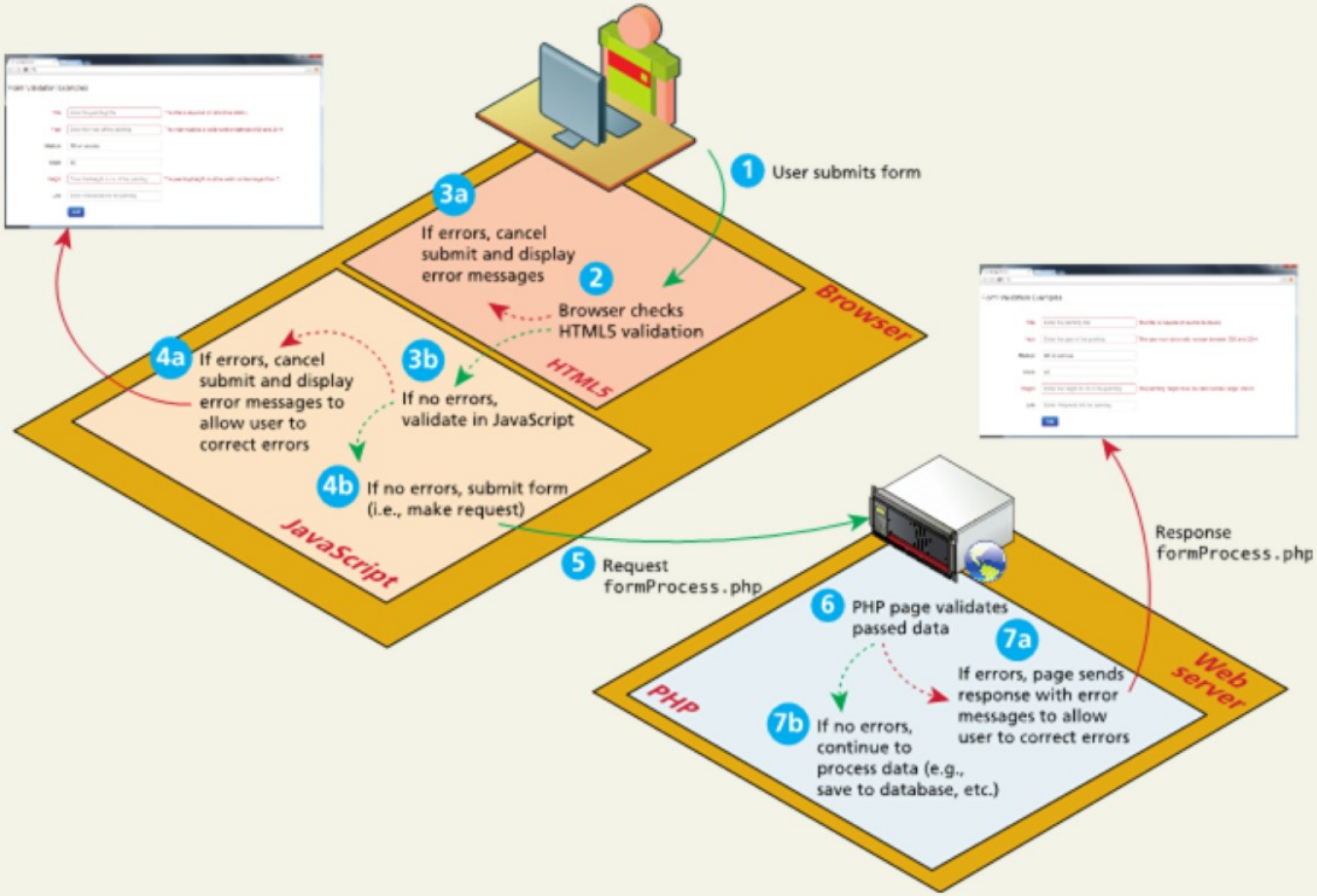
How to Reduce Validation Errors – use input masks



Chapter 15

1. What Are Errors and Exceptions?
2. PHP Error Reporting
3. PHP Error and Exception Handling
4. Regular Expressions
5. Validating User Input
6. Where to Perform Validation
7. Summary

Where to Perform Validation



Where to Perform Validation

Validation at the JavaScript Level

- Client process
- Can reduce server load
- Can be bypassed

Where to Perform Validation

Validation at the PHP Level

Validation on the server side using PHP is the most important form of validation and the only one that is absolutely essential.

Chapter 15

1. What Are Errors and Exceptions?
2. PHP Error Reporting
3. PHP Error and Exception Handling
4. Regular Expressions
5. Validating User Input
6. Where to Perform Validation
7. Summary

Summary

Key Terms

CAPTCHA

error

exception

expected error

fatal errors

literal

metacharacter

regular expression

spam bots

warnings

Summary

Questions?