

12.1 Types of Errors

1. Expected Errors

An error which routinely occurs during an application; egs User input errors (entering numbers when letters were expected etc).

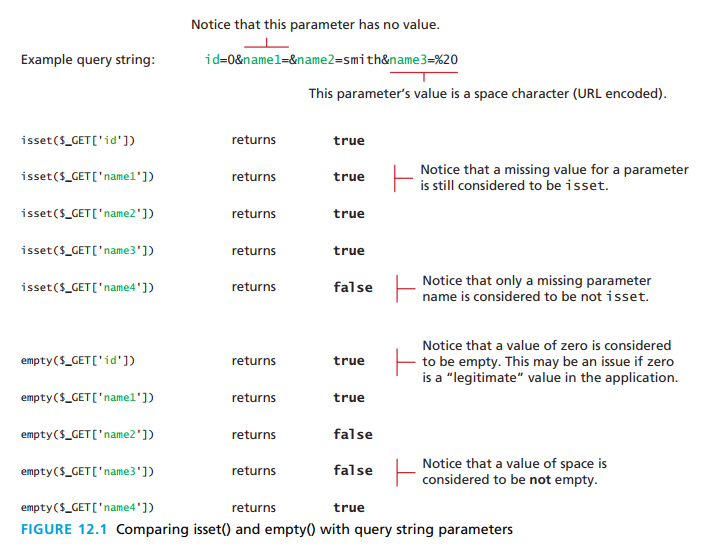
1. Warnings

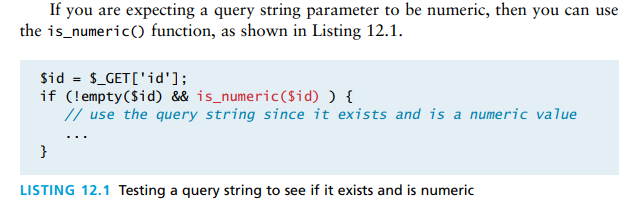
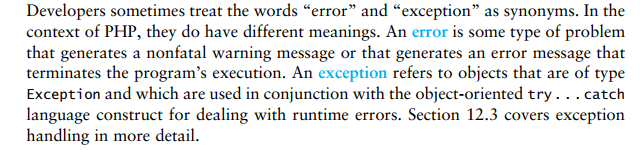
Problems which generate PHP warning message but will not halt execution of the page; this includes: calling a function without a required parameter.

1. Fatal Errors

Serious errors that will halt the execution of the page. Example, required input file missing or a database or field disappearing.

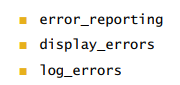
Isset() returns true if value is not null. Empty() returns true if value is null.

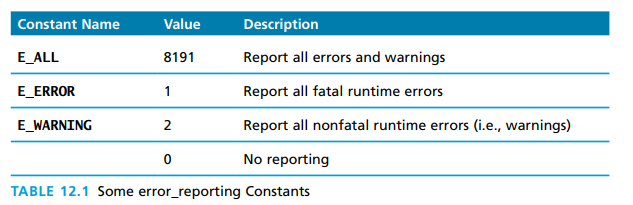


  
  
  
  
  
  
Exceptions  


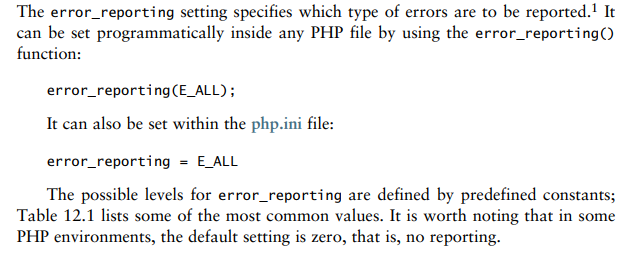
12.2 PHP error reporting

PHP has its own flexible and customizable system for reporting warnings and errors. These can be set programmatically or declaratively at design-time within the php.ini file.

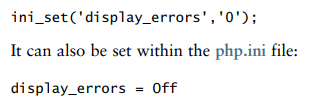
There are three main reporting flags:



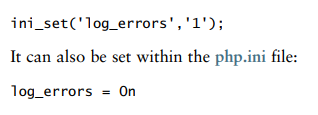
1. error\_reporting settings

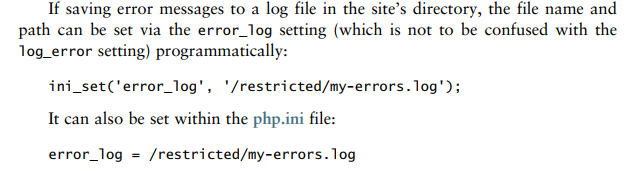
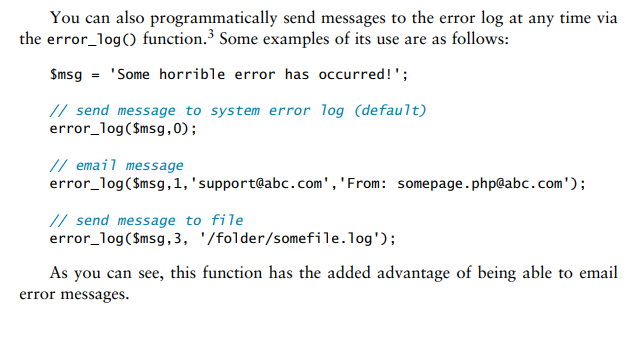


2. display\_errors

Determines if errors should be displayed in the browser. Can be set with ini\_set() function.

3. log\_error setting

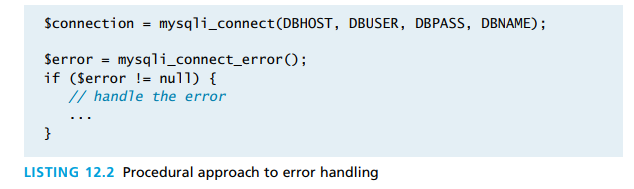
Determines whether error messages should or should not be sent to the server error log. Can be set via the ini\_set(function).

  
  
  
  
  
  
  
  
  
  
  
  
  
  
12.3 PHP error and Exception handling

Program execution will be interrupted if a fatal php error is not handled. There are two mechanisms to handle these types of errors.

1. Procedural Error Handling

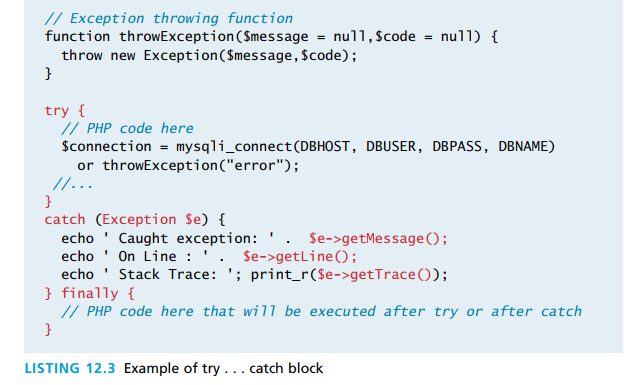
This is similar to chapter 11 procedural mysqli approach for accessing a database.



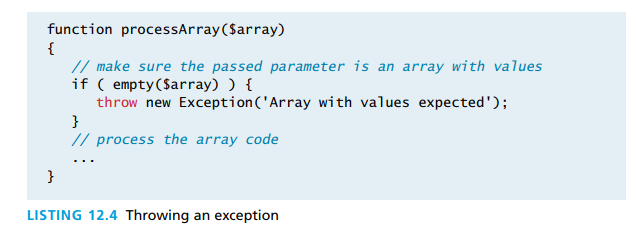
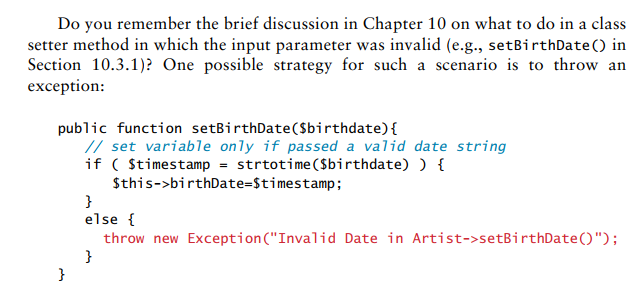
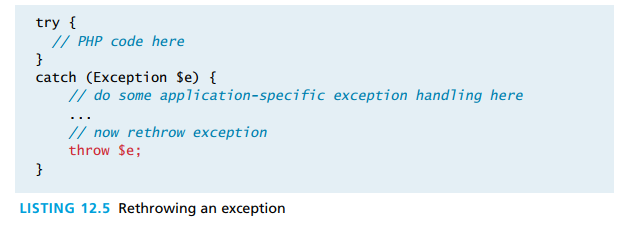
1. Object Oriented Exception Handling

When a run-time error occurs, PHP throws an exception; this exception can be caught and handled by a function, class or page. If it is not caught the PHP environment will handle it by terminating execution with an “Uncaught exception” message.

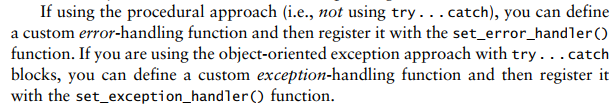
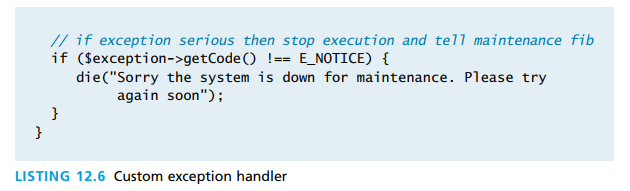
PHP uses try…catch programming construct to handle exceptions.



You can also use the throw keyword to handle exceptions:



1. Custom Error and Exception handling

  
  
  
  
You might want to hide some error message to the user [if you don’t want him to know some malicious error has occurred].   
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
12.4 Regular Expressions

Regular expression is a set of special characters that define a pattern. They are used with the intention of matching and manipulating texts; for example, to test whether or not user input matches a predictable set of patterns.

Regular expressions are a concise way to eliminate the conditional logic that would be needed to ensure input data follows a specific format.

1. Regular Expression Syntax

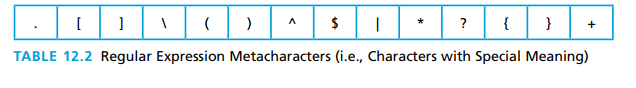
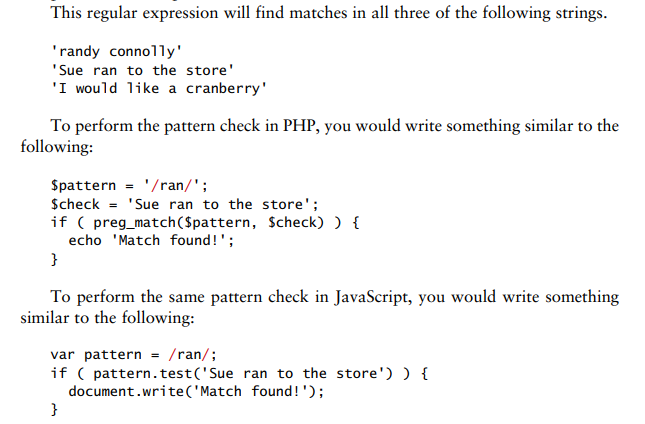
Consists of:

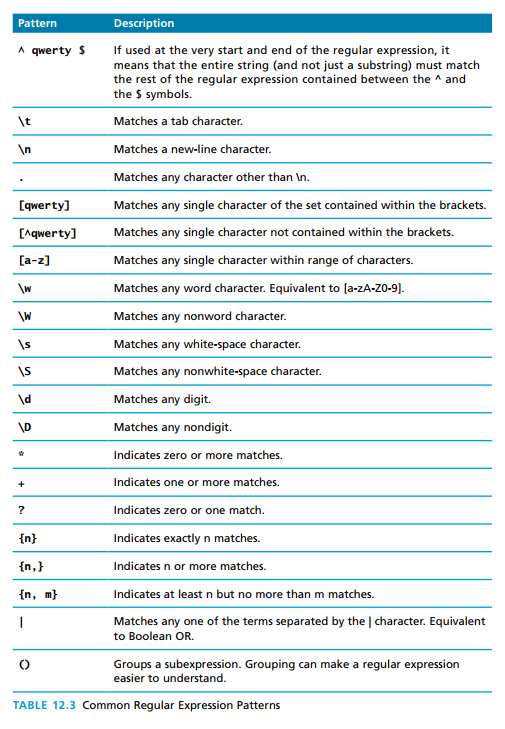
1. Literal

A character you wish to match in the target.

1. Metacharacter

It is a special symbol that acts as a command to the regular expression parser. To use metacharacter as a literal you must use the backslash prefix.





1. Extended Example

We can create our own regular expressions; for example, we could create a regular expression for that would match a north American number without the area code  
  
  
  
  
  
  
  
^ indicates the first

$ indicates the end

d indicates a digit

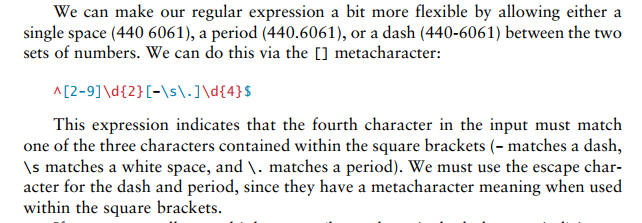
{3} or {4} indicates the length of the digits

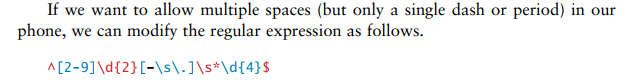
Example: 555-3333



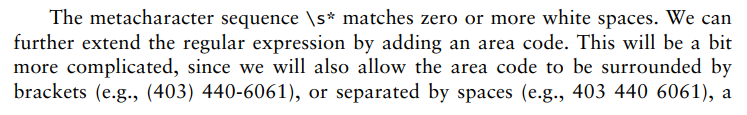
This example indicates that the first digit must be between 2-9 [not include zero or 1].

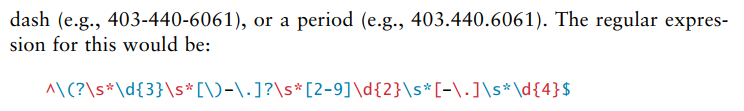
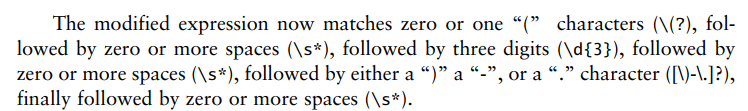
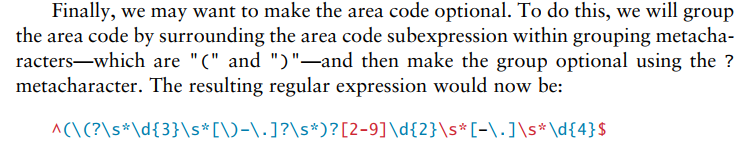
355-5555





/s\* allows for multiple white spaces.



  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Even though this may look frightening, this is much better than the general methods that we would use if we were using a function to the same thing; we can see this below in the example of a java-script method:

