What is an Exception?

An exception is an object that describes an error or unexpected behaviour of a PHP script.

Exceptions are thrown by many PHP functions and classes.

User defined functions and classes can also throw exceptions.

Exceptions are a good way to stop a function when it comes across data that it cannot use.

Throwing an Exception

The `throw` statement allows a user defined function or method to throw an exception. When an exception is thrown, the code following it will not be executed.

If an exception is not caught, a fatal error will occur with an "Uncaught Exception" message.

Lets try to throw an exception without catching it:

Example

```php
<?php
function divide($dividend, $divisor) {
  if($divisor == 0) {
    throw new Exception("Division by zero");
  }
  return $dividend / $divisor;
}

echo divide(5, 0);
?>
```

The result will look something like this:

```
Fatal error: Uncaught Exception: Division by zero in
C:\webfolder\test.php:4
```

```
Stack trace: #0 C:\webfolder\test.php(9):
divide(5, 0) #1 {main} thrown in C:\webfolder\test.php on line 4
```

The try...catch Statement

To avoid the error from the example above, we can use the `try...catch` statement to catch exceptions and continue the process.

Syntax

```
try {
  code that can throw exceptions
} catch(Exception $e) {
  code that runs when an exception is caught
}
```

Example

Show a message when an exception is thrown:

```
<?php
function divide($dividend, $divisor) {
  if($divisor == 0) {
    throw new Exception("Division by zero");
  }
  return $dividend / $divisor;
}

try {
  echo divide(5, 0);
} catch(Exception $e) {
  echo "Unable to divide.";
}
?>
```

The catch block indicates what type of exception should be caught and the name of the variable which can be used to access the exception. In the example above, the type of exception is `Exception` and the variable name is `$e`.

The try...catch...finally Statement

The `try...catch...finally` statement can be used to catch exceptions. Code in the `finally` block will always run regardless of whether an exception was caught. If `finally` is present, the `catch` block is optional.

Syntax

```
try {
  code that can throw exceptions
} catch(Exception $e) {
  code that runs when an exception is caught
} finally {
  code that always runs regardless of whether an exception was caught
}
```

Example

Show a message when an exception is thrown and then indicate that the process has ended:

```
<?php
function divide($dividend, $divisor) {
  if($divisor == 0) {
    throw new Exception("Division by zero");
  }
  return $dividend / $divisor;
}

try {
  echo divide(5, 0);
} catch(Exception $e) {
  echo "Unable to divide. ";
} finally {
  echo "Process complete.";
}
?>
```

Example

Output a string even if an exception was not caught:

```
<?php
function divide($dividend, $divisor) {
  if($divisor == 0) {
```

```
    throw new Exception("Division by zero");
  }
  return $dividend / $divisor;
}

try {
  echo divide(5, 0);
} finally {
  echo "Process complete.";
}
?>
```

The Exception Object

The Exception Object contains information about the error or unexpected behaviour that the function encountered.

Syntax

```
new Exception(message, code, previous)
```