Parse Error (Syntax Error):

1. Caused by a mistake in the PHP syntax.
2. Common examples include missing semicolons or curly braces.
3. Execution stops immediately when a parse error occurs.

```
echo "Hello, World"
```

Fatal Error:

1. Caused when PHP understands what you've written, but what you're asking it to do can't be done.
2. Common examples include calling an undefined function or class.
3. Execution stops immediately when a fatal error occurs.

```
echo myUndefinedFunction();
```

Warning Error:

1. Less severe than fatal errors.
2. They indicate something that's probably an error, but not so severe that execution must be halted.
3. An example might be including a file that doesn't exist using the include() function. The script will continue to execute after the warning.

```
 include("non_existent_file.php");
 echo "Hello";
```

Notice Error:

1. Less severe than warnings.
2. Indicate something that might be an error or might be perfectly fine, but it's worth investigating.
3. A common example is trying to access an undefined variable.

```
  $x = "Hello, World!";
  echo $y;
```

Deprecated Error:

1. These errors are thrown for functions or features that are outdated and will likely be removed in future versions of PHP.
2. An example might be using a function that has been deprecated in favor of a newer, better function.

```
$lat = 34.0522;
$lon = -118.2437;
$sunrise = date_sunrise(time(), SUNFUNCS_RET_STRING, $lat, $lon);
echo "Sunrise time in Los Angeles is: $sunrise";
```

Strict Error:

1. Indicate that you're using PHP in a way that may not be compatible with future versions.
2. They are suggestions for writing more robust code.
3. An example might be calling a non-static method statically.

```
class MyClass {
    function nonStaticMethod() {
        echo "nonStaticMethod";
    }
}
MyClass::nonStaticMethod();
```

Recoverable Fatal Error:

1. A fatal error that can be caught and handled using a custom error handler.
2. An example might be type hinting against a class and then passing an incorrect object type.

```
function myFunction(array $myArray) {
    echo $myArray[0];
}
myFunction("this is not an array");
```

User Errors:

1. You can trigger custom error messages using the `trigger_error()` function.
2. This allows you to define your own error levels like E_USER_ERROR, E_USER_WARNING, and E_USER_NOTICE.

```
if (1 !== 2) {
    trigger_error("One does not equal two!", E_USER_NOTICE);
}
```

#php