**PHP\_EOL**

====================

PHP\_EOL에서 뒤에 붙은 EOL은 end of line 의 약자 입니다. PHP\_EOL은 PHP의 내장 상수로 줄바꿈을 처리한다는 뜻입니다.

|  |
| --- |
| Echo “테스트”.PHP\_EOL; |

을 콘솔형태로 출력해보면 실제 줄바꿈이 이루어 지는 것을 확인해 보실 수 있습니다.

**클로저 외부변수**

====================

클로저는 외부의 있는 변수와 상호작용을 할 수 있습니다. 다음과 같이 클로저에 use 절을 사용하여 외부의 변수를 가지고 올 수 있습니다.

예제) Closure3.php

|  |
| --- |
| <?php  // 외부 변수 지정  $string = "Hello World";  $closure = function() use ($string) {  echo $string;  };  // 클로저 호출  $closure();  ?> |

**화면출력)**

Hello World

use 절을 통하여 상호작용 되는 변수의 값은 함수가 정의 된 시점부터 입니다.

예제) Closure4.php

|  |
| --- |
| <?php  $message = 'hello';  // No "use"  $closure = function () {  var\_dump($message);  };  echo $closure();  // $message 상속  $closure = function () use ($message) {  var\_dump($message);  };  echo $closure();  // 상속 된 변수의 값은 함수가 정의 된 시점부터입니다.  // 메시지 값을 변경해도 적용되지 않습니다.  $message = 'world';  echo $closure();  ?> |

**화면출력)**

C:\php-7.1.4-Win32-VC14-x86\Closure4.php:6:null

C:\php-7.1.4-Win32-VC14-x86\Closure4.php:12:string 'hello' *(length=5)*

C:\php-7.1.4-Win32-VC14-x86\Closure4.php:12:string 'hello' *(length=5)*

상호작용시 변수의 값을 직접 변수명으로 넣어 주는 것과 달리 참조 방식으로 전달할 수도 있습니다.

예제) Closure5.php

|  |
| --- |
| <?php  $message = '안녕하세요';  // 참조에 의한 상속  $example = function () use (&$message) {  var\_dump($message);  };  echo $example();  // 부모 스코프의 변수 값을 변경합니다.  // 변경된 값을 적용합니다.  $message = '환영합니다.';  echo $example();  // 클로저는 인수값을 같이 받아 들일 수도 있습니다.  $example = function ($arg) use ($message) {  var\_dump($arg . ' ' . $message);  };  $example("hello");  ?> |

화면출력)

C:\php-7.1.4-Win32-VC14-x86\Closure5.php:7:string '안녕하세요' *(length=15)*

C:\php-7.1.4-Win32-VC14-x86\Closure5.php:7:string '환영합니다.' *(length=16)*

C:\php-7.1.4-Win32-VC14-x86\Closure5.php:18:string 'hello 환영합니다.' *(length=22)*

**Goto 문법**

====================

goto 문법은 초기 개발언어부터 존재하는 제어문법 입니다. 어셈블리의 jump 와 같이 goto 명령을 만나게 되면 GOTO 라벨명이 있는 곳으로 프로그램의 제어권이 이동합니다.

goto 는 한 개의 실행 명령과 이동되는 라벨명 한쌍으로 구성이 됩니다.

즉

goto aaa:

…

…

aaa:

형태로 지정한 라벨로 이동을 합니다.

예) goto1.php

|  |
| --- |
| <?php  goto a;  echo 'Foo';    a:  echo 'Bar';  ?> |

화면출력)

Bar

위의 예를 보면 스크립트가 실행되자 마자 goto 명령을 만나 라벨 a: 로 이동을 합니다. 이동후에 echo 명령이 실행됩니다.

예) goto2.php

|  |
| --- |
| <?php  for($i=0,$j=50; $i<100; $i++) {  while($j--) {  if($j==17) goto end;  }  }  echo "i = $i";  end:  echo 'j hit 17';  ?> |

화면출력)

j hit 17

goto 명령을 사용 할 때 는 문법구조에 주의하여 사용을 해야 합니다. goto 명령은 동일한 스크립트 파일 안에 있어야 합니다. 다른 스크립트 파일로는 이동할 수 없습니다. 또한 함수나 메소드등을 탈출해서 다른 곳으로 이동할 수 없습니다.

또한 반복문 루푸나 switch 구문 안으로 이동을 할 수도 없습니다.

예) goto3.php

|  |
| --- |
| <?php  goto loop;  for($i=0,$j=50; $i<100; $i++) {  while($j--) {  loop:  }  }  echo "$i = $i";  ?> |

화면출력)

Fatal error: 'goto' into loop or switch statement is disallowed in

script on line 2

**자료형 연산자**

====================

자료형 연산자인 instanceof는 PHP변수의 클래스 객체 여부를 확인할 수 있습니다.

예제) instanceof-1.php

|  |
| --- |
| <?php  class MyClass  {  }  class NotMyClass  {  }  $a = new MyClass;  var\_dump($a instanceof MyClass);  var\_dump($a instanceof NotMyClass);  ?> |

화면출력)

C:\php-7.1.4-Win32-VC14-x86\instanceof-1.php:11:boolean true

C:\php-7.1.4-Win32-VC14-x86\instanceof-1.php:12:boolean false

또한 instanceof는 클래스의 부모 클래스 상속여부도 확인을 할 수 있습니다.

예제) instanceof-2.php

|  |
| --- |
| <?php  class ParentClass  {  }  class MyClass extends ParentClass  {  }  $a = new MyClass;  var\_dump($a instanceof MyClass);  var\_dump($a instanceof ParentClass);  ?> |

화면출력)

C:\php-7.1.4-Win32-VC14-x86\instanceof-2.php:12:boolean true

C:\php-7.1.4-Win32-VC14-x86\instanceof-2.php:13:boolean true

또한 instanceof는 클래스의 인터페이스 적용여부도 확인을 할 수 있습니다.

예제) instanceof-3.php

|  |
| --- |
| <?php  interface MyInterface  {  }  class MyClass implements MyInterface  {  }  $a = new MyClass;  var\_dump($a instanceof MyClass);  var\_dump($a instanceof MyInterface);  ?> |

화면출력)

C:\php-7.1.4-Win32-VC14-x86\instanceof-3.php:12:boolean true

C:\php-7.1.4-Win32-VC14-x86\instanceof-3.php:13:boolean true