jinyPHP 1권 – 03. 기본구분

**3 기본 구문**

**3.1 PHP 기본 문법**

PHP는 스크립트 언어입니다.

스크립트언어는 또 다른 말로 인터프리터 언어라고도 말할 수 있습니다. 즉 프로그램이 실행될 때마다 해당 언어 소스를 컴퓨터가 이해할 수 있는 기계어 코드로 변환하여 처리하는 언어입니다. 대표적으로는 베이직, 자바, PHP 등이 있습니다.

**PHP는 서버 사이드 언어**

서버 사이드 언어란 프로그램을 실행하는 주체가 사용자 각각의 컴퓨터에서 동작하는 것이 아니라 주로 서버에서 프로그램이 실행된다는 의미입니다. 또한, 서버는 이렇게 실행된 결과물을 각각의 접속된 클라이언트 PC에 내용을 출력하는 역할을 합니다. 이런 점에서 PHP는 대표적인 서버-클라이언트 기반의 웹 서비스 용도로 많이 사용합니다.

PHP가 서버 사이드 언어라고 해서 꼭 서버에서만 동작하는 것은 아닙니다. 서버란 추상적인 의미일 뿐 일반적인 개인용 컴퓨터에서도 작동이 됩니다. 개인 컴퓨터에서 서비스 개발한 후에 소스를 서버로 옮겨서 서비스하면 됩니다.

**프로그램 코딩**

코딩이란 로직을 설계하고 설계한 로직을 프로그래밍 언어의 문법에 따라 일정한 양식으로 작성하는 명령 집합을 생성하는 작업입니다. 또한 이러한 코딩으로 작성된 프로그램은 개발자가 지시한 명령에 따라 데이터를 처리하고 결과를 출력합니다.

프로그램을 개발하다 보면 다양한 오류가 발생할 수 있습니다. 오류는 크게 두 가지로 구분할 수 있습니다.

* **구문 오류**: 코딩을 하다 보면 콤마, 세미콜론 등이 틀리거나 오탈자로 인한 문법 오류가 나는 경우가 있습니다. 이런 경우는 언어에서 오류를 알려주기 때문에 오류 메시지를 잘 보면 좋습니다.
* **논리 오류**: 프로그램을 작성하다가 설계상의 오류가 발생한 경우입니다. 이런 경우 설계 로직을 다시 한번 확인하는 것이 좋습니다.

**3.1.1 PHP 프로그램 작성하기**

PHP를 통해 프로그램을 작성하는 것은 매우 자유롭습니다. 특정 개발 환경 프로그램을 사용해서 코딩하는 것이 아니라 우리가 일반적으로 사용을 하는 텍스트 에디터를 이용하여 자유롭게 작성을 할 수 있습니다. 그냥 선호하는 에디터를 선택하여 사용하면 됩니다.

**코딩:**

프로그램 소스를 작성하는 행위

**에디터의 종류:** 프로그래머는 보통 자신이 선호하는 텍스트 에디터가 하나씩은 있습니다. 필자의 경우 서브라임(sublime)이나 아톰(atom) 에디터를 선호합니다. 또한 이 프로그램은 무료로 자유롭게 다운로드하여 사용 가능합니다.

**PHP 코드를 상징하는 기호 <?와 ?>**

프로그램 소스상에서 PHP 인터프리터가 파일의 전부를 PHP 코드로 인식하지는 않습니다. 소스 파일을 읽을 때 <? 또는 <?php 기호를 만나는 순간부터 PHP 소스로 인식합니다. 또한 ?>를 만나면 PHP 소스가 끝나는 시점으로 해석을 종료합니다.

이런 자유로운 프로그램 코딩 방식은 PHP를 다른 언어에 삽입하여 두 가지 이상의 언어와 혼합해서 사용 가능하다는 장점이 있습니다. 주로 HTML 소스 안에 PHP 소스 코드를 삽입하거나 또는 HTML 소스 안에서 자바스크립트 코드와 PHP 코드를 혼합하여 프로그래밍이 가능할 수도 있습니다.

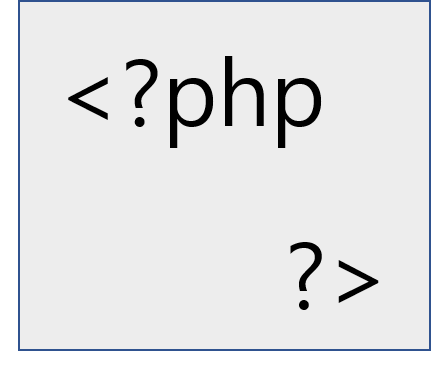
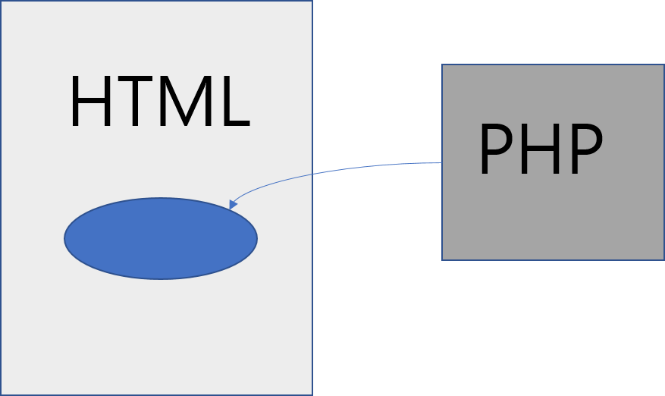
이런 프로그램 코드의 강점과 달리 유지보수 측면을 고려하여 별도로 PHP 코드만 분리하는 경우도 있습니다. 보통 프레임워크가 MVC 패턴에서 view 처리 부분이 이와 유사합니다. PHP는 HTML등 다른 언어와 같이 사용해도 되고 독립적으로 사용해도 괜찮습니다. 모두 정상적으로 잘 동작합니다.

다음 PHP 언어를 HTML마크업과 같이 사용하는 예입니다.

**예제 파일** hello-01.php

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1> PHP 페이지를 만들어 봅니다.</h1>  **<?php**  **echo "Hello World!";**  **?>**  </body>  </html> |

위의 예제를 보면 HTML 코드 안에 PHP 코드가 삽입되어 있습니다.



또는 PHP코드만 독립적으로 사용을 할 때는 다음과 같이 사용해도 됩니다

|  |
| --- |
| <?php  코드…  ?> |

하나의 프로그램 소스 안에서 <?php와 ?>은 한번만 사용을 해야 되는 것은 아닙니다. 그때마다 필요할 때 PHP코드들을 분리해서 삽입이 가능합니다.

아래 예제는 하나의 HTML 파일에 2개의 php 코드 그룹을 넣은 예제입니다.

**예제 파일** hello-02.php

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1> PHP 페이지를 만들어 봅니다.</h1>  **<?php**  **echo "Hello World!";**  **?>**  <h2> 두 번째로 PHP 코드를 삽입을 합니다. </h2>  **<?php**  **echo "good morning!";**  **?>**  </body>  </html> |

위처럼 HTML 마크업 안에서 2개의 PHP코드를 그룹 분할하여 삽입하는 기능은 PHP 코드가 다른 언어와 연결하여 프로그래밍하는 데 많은 도움을 줄 수 있습니다.

**php 종료 태그 생략**

PHP는 종료 태그 “?>”로 끝나는 경우에 한하여 다음과 같이 태그를 생략할 수 있습니다. 하지만 PHP 코딩 스타일 PSR-2 에서는 PHP 소스의 마지막에 종료 태그 ?>를 생략해서 쓰는 방법을 권장하고 있습니다. 종료 태그 ?>를 생각하게 되면 예상치 않은 출력 오류를 방지할 수도 있습니다.

예제파일 hello-03.php

|  |
| --- |
| <?php  echo "Hello World!"; |

위의 예제 표현은 마지막에 종료 태그 ?>를 삭제한 것입니다. 태그를 삭제해도 PHP 실행에는 지장이 없습니다.

**3.1.2 PHP 확장자**

모든 파일은 대부분 확장자(extension) 기호를 가지고 있습니다. 소스 파일을 열어보지 않고도 작성한 파일이 PHP 언어로 작성한 것인지를 쉽게 확인할 수 있도록 PHP 언어 또한 확장자 기호가 있습니다. PHP로 작성하는 프로그램은 대부분 **.php**라는 확장자로 구분합니다.

하지만 꼭 확장자를 .php로 써야만 하는 것은 아닙니다. .html 또는 .phtml 등 다양한 확장자를 쓰기도 합니다. 즉, PHP는 확장자로 인해 프로그램 실행에 영향받지 않습니다. .htm 확장자를 가진 HTML 파일에서도 <?php ~~ ?> 기호를 통해 PHP 프로그램 소스를 삽입할 수 있습니다.

참고로 .php 확장자 의미와 지정은 환경 설정 및 아파치와 같은 서버 프로그램에서 설정한 값을 따릅니다.

**03.1.3 HTML, CSS, Javascript**

PHP는 웹 서비스 개발 용도로 많이 사용을 하는 서버 사이드 언어입니다. 따라서 위의 예제처럼 PHP 소스 코드 이외에 약간의 HTML, CSS, Javascript 내용이 함께 들어가는 경우가 있습니다.

이 책에서는 이와 관련하여 별도의 설명을 하지 않습니다. 관련 정보는 미리 습득하면 이 책을 좀 더 쉽게 이해하는 데 도움이 될 것입니다.

**3.2 명령어 구분: 세미콜론(;)**

PHP는 C 언어나 자바처럼 프로그래밍 언어 중에서도 고급 언어에 속합니다.

**고급 언어란** 컴퓨터가 이해하는 기계어 나 어셈블리어처럼 마이크로프로세서 중심의 언어로 작성되는 것이 아닌, 사람이 일반적인 언어처럼 쉽게 문장을 이해할 수 있는 형태의 언어 구조를 말합니다. 대표적인 고급 언어로는 C 언어, 베이직, PHP, 자바 등이 이에 속합니다.

기본적으로 프로그램은 소스상의 한 줄 단위로 명령어를 읽어서 컴퓨터에서 해석하고 동작합니다. 하지만 프로그래밍 언어가 고급 언어로 넘어오면서 한 줄로 다 표현할 수 없는 경우가 많아지고, 가독성이 떨어지는 문제가 있습니다. 이런 경우 한 줄 단위 명령처럼 하나의 명령의 끝을 표기해야만 컴퓨터가 한 개의 명령문을 이해할 수 있습니다.

이런 표기를 대부분의 언어에서는 세미콜론(;)을 이용하여 처리합니다. 즉 하나의 프로그램 문장이 시작하여 세미콜론을 만나게 되면 컴퓨터는 하나의 명령문으로 인식을 하고 처리합니다.

여기서 주의할 점은 많은 초보자 개발자들이 세미콜론(;)과 콜론(:)을 헷갈린다는 것입니다. 만일 소스상에서 세미콜론을 빼먹거나 잘못 입력했을 경우에는 프로그램 오류로 동작을 하지 않습니다. 또는, 프로그램 전체가 동작하지 않을 수 있습니다.

다음은 Hello World!라는 문장을 세 번 출력을 하는 간단한 PHP 프로그램입니다. 각각의 명령어 문장 뒤에는 세미콜론(;)이 붙어서 한 개의 명령문을 구분하고 있습니다.

예제파일:hello-04.php

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>    <?php  ECHO "Hello World!<br>";    echo "Hello  World!<br>";    ecHo "Hello  World!  <br>";  ?>    </body>  </html> |

첫 번째 Hello World!는 한 줄에 모두 쓰여 있어서 한 줄 명령과 헷갈릴 수도 있을 것입니다. 하지만 두 번째와 세 번째 명령을 보면 하나의 명령을 두 줄이나 세 줄로 표시해서 출력할 때 명령의 끝을 세미콜론(;)을 통해 다수의 라인으로 명령을 작성할 수 있습니다.

**명령의 구분자는 왜 중요할까요?** 그리고 명령 구분자 세미콜론(;)을 넣지 않으면 왜 프로그램이 실행되지 않을까요? 고급 언어의 경우 컴파일/인터프리터라는 처리 후에 컴퓨터가 인지되는 프로그램으로 변경되기 때문입니다.

고급 언어는 앞에서 언급한 것과 같이 컴퓨터가 이해를 하지 못하는 언어 구조입니다. 사람이 좀 더 쉽게 이해할 수 있도록 만든 가상의 언어입니다. 그래서 고급 언어는 실행 전에 컴파일러/인터프리터라는 프로그램이 고급 언어로 작성된 소스를 읽어 세미콜론(;)으로 구분을 하고 각각의 작성된 내용을 분석하여 기계어로 만듭니다.

그래서 명령어의 구분자인 세미콜론(;)을 입력하지 못하면 컴파일러/인터프리터가 작성된 소스를 구분할 수 없고, 잘못 분석하여 처리하기 때문에 프로그램이 동작을 하지 않습니다.

**3.2.1 컴파일러란?**

우리가 작성한 코드는 컴퓨터에서 실행할 수 있는 코드로 변환되어 동작하게 됩니다. 이러한 변환 작업을 컴파일한다고 표현합니다. 컴파일 작업을 한 번하게 되면 소스 코드 이외에 컴퓨터 실행 코드로 된 기계어 파일이 추가로 생성됩니다. 실제적인 프로그램 동작은 이 기계어 파일이 됩니다.

만일 소스를 수정하게 되면 다시 컴파일 과정을 거쳐서 실행되는 기계어 파일을 다시 생성해야만 합니다. 컴파일된 기계어 파일은 한 번의 언어 변환을 했기 때문에 빠른 실행이 가능합니다. 하지만 매번 소스 변경할 때마다 컴파일해야 하는 불편함이 있습니다. 대표적인 컴파일 언어로는 C언어가 있습니다.

**3.2.2 인터프리터란?**

이와 반대로 컴파일 작업을 하지 않고 소스를 실행할 때마다 해석하여 기계어 코드로 실행하는 방법을 인터프리터라고 합니다. 인터프리터는 프로그램이 실행될 때마다 소스를 해석하고 이와 관련된 기계어 코드를 임시로 생성하여 실행합니다. 컴파일된 기계어 코드보다는 다소 느리지만 소스를 변경할 때마다 컴파일을 하지 않기 때문에 빠른 개발이 가능합니다.

인터프리터 언어로 제작된 프로그램은 소스가 같이 공개 배포되기 때문에 작성한 코드를 암호화할 수 없습니다. 인터프리터 언어는 보안상 중요하지 않고 간단한 프로그램 제작에 많이 사용합니다. 또한 공개 오픈소스 언어로 많은 인기를 얻고 있습니다. 대표적으로는 베이직, PHP, 자바스크립트, 파이썬 등이 있습니다.

**3.3 주석문**

모든 프로그램 개발 언어는 주석문이라는 기능이 있습니다. 주석은 프로그램 언어에서 개발자가 소스상에 설명을 적는 역할을 합니다. 방대한 프로그램 소스를 여러 사람이 공유하면서 상호 이해하거나 아니면 자신이 작성한 코드를 그때의 기억을 쉽게 떠올리기 위해 사용하는 기능이 주석 기능입니다.

주석 기능은 설명을 적기 위해 도입되었지만 실제 상황에서는 설명만이 아니라 잠시 소스 코드의 실행을 배제하는 용도로도 많이 사용합니다. 또한 주석 문법은 대부분의 언어가 비슷하기 때문에 한 개의 언어로 학습을 해둔다면 다른 언어를 배우는 데도 많은 도움이 됩니다.

주석은 크게 두 가지 방식을 제공합니다. 첫 번째는 소스를 읽었을 때 한 줄 한 줄을 주석으로 처리하는 한 줄 주석 처리문과 여러 줄을 한꺼번에 처리하는 방법이 있습니다.

**3.3.1 한 줄 주석 처리**

한 줄 주석 처리는 프로그램 소스상에서 한 줄만 주석으로 처리합니다. 한 줄을 주석으로 처리하기 위해서는 주석을 처리하고자 하는 위치에서 백슬래시(/) 기호를 두 번 연속해서 넣으면 됩니다. 즉, 주석을 처리하고자 하는 위치에 // 기호를 넣으면 됩니다.

한 줄 주석 처리 기호의 시작 위치는 중요하지 않습니다. 기호를 삽입하는 바로 다음 문자부터 한 줄의 끝을 만나는 지점까지 모두를 주석으로 처리합니다.

예제파일: comment-01.php

|  |
| --- |
| <?php  // <--한 줄의 시작부터 주석이 시작될 수 있습니다.  $name = "hello world!"; // <-- PHP코드 뒤부터 한 줄 처리 주석을 시작할 수 있습니다.  ?> |

한 줄 주석을 사용하는 습관은 프로그래머마다 다릅니다. 하지만 필자의 경우 대부분 첫 줄부터 시작하는 주석은 해당 코드의 설명을 주로 기입하고, 코드 뒤에 입력한 주석은 코드의 상태나 부연 설명 등을 위해 주로 사용합니다.

**3.3.2 여러 줄 주석 처리**

여러 줄 주석 처리 기능은한 줄 처리 주석과 달리 여러 줄의 내용을 한꺼번에 주석으로 처리하고자 할 때 사용하는 방법입니다. 특정 블록 영역을 주석으로 처리할 때 매우 편리합니다. 여러 줄을 주석으로 처리하는 방법은 **/\*로 시작해서 \*/**로 블록을 정하면 됩니다. 해당 기호로 감싸인 내용은 프로그램 소스상에서 주석으로 처리되어 실행되지 않습니다.

다음 예제에서 여러 줄 주석 처리 방법을 확인할 수 있습니다.

예제파일: comment-02.php

|  |
| --- |
| <?php  // This is a single-line comment    /\*  This is a multiple-lines comment block  that spans over multiple  lines  \*/    // You can also use comments to leave out parts of a code line  $x = 5 /\* + 15 \*/ + 5;  echo $x;  ?> |

여러 줄 주석 처리 **/\* \*/는 중첩되어 사용할 수 없습니다.** 즉, /\* \*/로 주석 처리된 블록 안에서 또 다른 /\* \*/ 주석 블록을 삽입할 수 없다는 뜻입니다. 이런 경우 PHP는 에러를 발생하고 실행이 중단됩니다.

**여러 줄 주석 처리의 잘못된 예)**

|  |
| --- |
| /\*  여러 줄 주석입니다.  /\*  서브로 여러 줄 주석은 삽입을 할 수 없습니다.  \*/  주석의 끝입니다.  \*/ |

다중 주석 처리 문 안에는 한 줄 주석을 삽입할 수 있습니다.

|  |
| --- |
| /\*  여러 줄 주석입니다.  // 중간에 한 줄 주석문은 삽입이 가능합니다.  //  주석의 끝입니다.  \*/ |

오픈소스 등 공개된 소스를 다운로드해서 보면 수많은 줄의 설명 주석을 보게 됩니다. 이처럼 주석문을 잘 사용을 하면 소스 오류를 테스트하고 다른 사람과 개발에 대한 정보를 소스 코드에 설명을 삽입하여 공유할 수 있습니다.

주석을 잘 작성하는 것은 매우 중요합니다. 오래되고 경험이 많은 개발자일수록 주석문을 통해 소스에 대한 설명을 많이 기재합니다.

**3.4 화면 출력**

앞으로 PHP를 본격적으로 공부하기에 앞서 많은 예제를 만나보게 될 것입니다. 또한 예제 소스들은 결과내용을 출력하는 화면 출력 함수들을 많이 사용합니다. 그럼 점에서 이해가 어렵더라도 뒤에서 설명할 결과를 확인하는 과정과 화면 출력에 대한 함수 일부분에 대해서 미리 살펴보겠습니다.

PHP에서 화면 출력 함수는 아주 다양합니다. 하지만 여기서는 지면상 모든 내용을 다 설명할 수는 없으므로 몇 가지 화면 출력 명령에 대해서 이야기하겠습니다. 대표적인 명령으로는 echo와 print 가 있습니다.

**3.4.1 ECHO**

PHP에서는 강력하면서 간단하게 사용할 수 있는 echo 명령이 있습니다. echo 명령은 이후에 표시되는 값을 화면에 출력하는 명령입니다. echo는 명령어도 될 수 있고 함수도 될 수 있습니다. echo는 ()를 사용해도 되고 생략해도 됩니다.

**명령어 문법)**

echo 출력한내용;

echo (출력할내용);

위의 두 가지 표현을 모두 사용을 할 수 있습니다. echo 명령문은 상수 값, 직접 입력되는 문자열 및 각종 변수의 값을 출력할 수 있습니다. 또한 여러 개의 출력 내용을 연결하고 연산된 값을 바로 출력하는 등 강력한 화면 출력 기능을 제공합니다.

**echo 실습1)**

문자열을 화면에 출력할 수 있습니다. echo 뒤에 직접 문자열 값을 입력하여 화면에 출력할 수 있습니다.

예제파일: echo-01.php

|  |
| --- |
| <?php  echo "I love PHP!";  ?> |

**echo 실습2)**

변수의 값을 화면에 출력할 수 있습니다. 문자열 "jinyPHP.com"을 $txt 변수에 저장하고 저장된 $txt 변수의 값을 화면에 출력합니다. 변수의 자세한 개념은 다음 장에서 설명합니다.

예제파일: echo-02.php

|  |
| --- |
| <?php  $txt = "jinyPHP.com";  echo $txt;  ?> |

**echo 실습3)**

문자열과 변수 값을 결합하여 화면에 출력할 수 있습니다. 직접 입력된 문자열과 변수에 저장된 문자열을 연결하는 연산자인 점(.)을 통해 연결 후 화면에 출력합니다. 연산자에 대한 자세한 개념은 다음 장에서 설명합니다.

예제파일: echo-03.php

|  |
| --- |
| <?php  $txt = "jinyPHP.com";  echo "I love " . $txt . "!";  ?> |

**echo 실습4)**

연산 결과 값을 화면에 출력할 수 있습니다. 변수 $x와 $y의 변수 값을 연산 후 바로 화면 출력이 가능합니다.

예제파일: echo-04.php

|  |
| --- |
| <?php  $x = 5;  $y = 4;  echo $x + $y;  ?> |

**3.4.2 print**

PHP는 echo 출력문 이외에 다른 언어에서 쉽게 접해볼 수 있었던 print 명령도 같이 지원합니다. print명령문 또한 화면에 출력하는 기능을 합니다. print 명령어는 echo와 같이 ()를 사용해도 되고 생략해도 됩니다.

**print 실습1)**

print() 함수를 이용하여 문자열을 화면에 출력할 수 있습니다. 사용법은 echo와 비슷합니다.

예제파일: print-01.php

|  |
| --- |
| <?php  print "<h2>PHP is enjoy!</h2>";  print "Hello jiny world!<br>";  print "I'm about to learn PHP!";  ?> |

**print 실습2)**

변수의 값을 화면에 출력을 할 수 있습니다. 문자열 "jinyPHP.com"을 $txt변수에 저장하고, 저장된 $txt 변수의 값을 화면에 출력합니다.

예제파일: print-02.php

|  |
| --- |
| <?php  $txt = "jinyPHP.com";  print $txt;  ?> |

**print 실습3)**

문자열과 변수 값을 결합하여 화면에 출력할 수 있습니다. 직접 입력된 문자열과 변수에 저장된 문자열을 연결 연산자인 점(.)을 통하여 연결 후 화면에 출력합니다.

예제파일: print-03.php

|  |
| --- |
| <?php  $txt = "jinyPHP.com";  print "I love " . $txt . "!";  ?> |

**print 실습4)**

연산 결과 값을 화면에 출력할 수 있습니다. 변수 $x와 $y의 변수 값을 연산 후 바로 화면 출력이 가능합니다.

예제파일: print-04.php

|  |
| --- |
| <?php  $x = 5;  $y = 4;  print $x + $y;  ?> |

**3.4.3 echo VS print**

echo와 print 명령은 서로 유사한 기능을 하는 명령어입니다. 하지만 두 기능을 엄밀히 구분해 본다면 반환 값입니다. 반환 값은 print가 함수형으로 반환 처리를 한다는 것입니다. print는 명령 실행 후 true (1) 값을 반환하지만 echo는 그냥 화면에 출력만 합니다. 함수에 대한 자세한 개념은 다음 장에서 설명합니다.

**예제 파일** print-05.php

|  |
| --- |
| <?php  if ($success = print("hello World!")) {  echo "출력 성공 ".$success;  } else {  echo "출력 실패 ".$success;  }  ?> |

**결과)**

hello World!출력 성공 1

위의 예제는 print() 함수의 반환 값 성질을 이용한 예입니다. print 내장 함수를 통해 문자열을 출력하고 반환 값을 $success 변수에 저장합니다. $success 변수의 논리 값을 비교하여 출력 성공 메시지를 함께 출력합니다. 조건 문법 if의 자세한 개념은 다음 장에서 설명합니다.

**03.4.4 here document**

here document는 PHP 스크립트 안에서 긴 다중 라인의 문자열을 출력하거나 대입할 때 쓸 수 있는 문법입니다. 다른 말로 Newdoc 스타일이라고도 합니다.

<<< 직후의 문자열은 끝을 나타내는 문자열로 터미네이션 ID라고 합니다. 터미네이션 ID로는 EOF, EOL, EOT, END 등이 있습니다.

예제파일: newdoc-01.php

|  |
| --- |
| <?php  $text = <<<EOL  안녕하세요  EOL;  echo $text;  ?> |

문자열을 표현하는 따옴표 없이 화면에 출력할 수 있습니다.

예제파일: newdoc-02.php

|  |
| --- |
| <?php  echo <<<END  안녕하세요  END;  ?> |

변수 없이 바로 출력합니다.

Nowdoc 스타일의 종료 문자는 앞에 공백이 있으면 안 됩니다. 이러한 스타일은 변수나 특수문자, 큰따옴표, 작은따옴표 등을 섞어서 사용할 수 있습니다.

**3.5 PHP 실행**

간단하게 PHP의 문법적 구조와 표기법에 대해서 설명했습니다. 이렇게 작성한 PHP소스를 실행해 보겠습니다.

이전 PHP 설치에서 간단하게 테스트 페이지 실행에 대한 예가 있었습니다. PHP는 직접 콘솔을 통해 실행할 수도 있고 웹 서버를 통해 스크립트를 실행할 수도 있습니다. 웹 결과 화면은 웹 브라우저로 확인할 수 있습니다.

또한 계속 설명하듯이 PHP는 서버 사이드 웹 스크립트입니다. 작성한 php 파일을 웹 서버에 올려 놓으면 아파치와 같은 웹 서버가 php 스크립트 파일을 해석하여 웹 사이트로 출력합니다. 출력되는 웹 사이트는 PC에서 웹 브라우저를 통해 결과를 확인할 수 있습니다.

**3.5.1 도메인 접속**

PHP를 웹 서버를 통해 실행하기 위해서는 실행 주소가 필요합니다. 만일 PHP를 실행하는 접속 서버가 도메인 주소가 있는 경우 아래와 같이 서버의 도메인과 스크립트 파일 경로를 적으면 됩니다.

**실행 접속 )**

|  |
| --- |
| http:// 도메인 / 스크립트파일.php |

도메인을 접속하기 위해서는 사전에 유효한 도메인을 보유하고 있어야 합니다. 도메인은 도메인 전문 회사를 통해 1년 단위로 구입할 수 있습니다. 또한 구입한 도메인을 웹 서버(아파치)에서 서비스 가능하도록 환경 설정되어 있어야 합니다.

보다 자세한 도메인 및 웹 서버 구축에 대해서는 관련 서적을 참고하기 바랍니다.

**3.5.2 IP접속**

웹 서버는 서비스를 위해 인터넷에 접속 및 이와 관련한 IP 주소를 가지고 있습니다. 아직 도메인을 보유하지 않고 있다면 서버의 IP 주소를 통해 웹 서버에 접속할 수 있습니다.

또한 IP 주소는 도메인을 가리키는 실제적인 접속 주소입니다. 만일 PHP를 실행하는 접속 서버가 도메인이 없는 경우 IP 주소를 입력하여 아래와 같이 서버의 도메인 과 스크립트 파일 경로를 적으면 됩니다.

**실행 접속 )**

|  |
| --- |
| http:// IP주소 / 스크립트파일.php |

위의 예처럼 브라우저 주소창에서 IP주소를 직접 입력하여 실행할 수도 있습니다.

**3.5.3 localhost**

만일 PHP를 서버가 아닌 개인 PC에서 실행할 경우 도메인이나 서버 IP 접속을 할 수 없습니다. 또는 보유한 도메인 주소가 없는 경우에는 localhost 이름을 통해 php가 실행되어 있는 웹 서버로 접속이 가능합니다.

localhost는 127.0.0.1의 IP 주소 값을 가집니다.

**실행 접속 )**

|  |
| --- |
| http:// localhost / 스크립트파일.php |

|  |
| --- |
| http:// 127.0.0.1 / 스크립트파일.php |

**localhost 라는 말은 자기 자신의 컴퓨터를 지칭하는 이름입니다.** 서버나 개인용 PC 등 네트워크에 접속되어 있는 대부분의 컴퓨터는 localhost 접속 및 IP 주소를 가지고 있습니다.

따라서 개인 PC나 개인 개발 환경의 경우 localhost 이름을 이용하면 자신의 컴퓨터에 쉽게 접속을 할 수 있습니다.

**3.5.4 콘솔 실행**

PHP는 클라이언트 웹 브라우저를 통해 실행 및 결과를 확인하는 것 이외에 서버, 터미널 콘솔 창에서 실행 결과를 확인할 수 있습니다.

**실행 방법)**

|  |
| --- |
| # php 실행 파일.php |

위와 같이 콘솔 창에서 실행하면 바로 결과를 콘솔 창에서 확인할 수 있습니다.

콘솔 창과 웹 브라우저로 결과를 확인하는 것은 약간의 차이점이 있을 수 있습니다. 콘솔 창에서 출력의 다음 줄은 ‘\n’으로 표기하지만 웹 브라우저는 다음 줄을 <br> 태그로 처리합니다.

대부분의 소스들은 웹 브라우저에서 확인하는 것을 기준으로 다음 줄 표시를 <br> 태그를 사용합니다. <br> 태그는 다음 줄을 의미한다고 생각하면 됩니다.

PHP 콘솔을 이용하면 몇 가지 장점이 있습니다.

* PHP 스크립트 실행 파일을 백그라운드로 실행할 수도 있습니다.
* 스케줄링 기능을 통해 주기적으로 PHP를 실행할 수도 있습니다.

**3.6 PHP 버전 확인**

PHP 설치 설명에서 잠시 아래와 같은 예제를 본 적이 있을 것입니다. PHP는 현재 설치되어 있는 PHP의 환경 설정 및 버전을 웹 브라우저 접속을 통해 확인할 수 있는 특수한 함수를 제공합니다.

예제파일: phpinfo.php

|  |
| --- |
| <?php  phpinfo();  ?> |

위와 같이 phpinfo();라는 함수 한 줄을 코드에 넣으면 PHP는 자신의 환경 설정 정보를 웹 페이지 형태로 화면에 출력합니다.

phpinfo()는 PHP버전 과 설정 정보, HTTP 헤더, 확장 모듈 및 사전 정의된 변수 등도 함께 출력됩니다.

주의할 점은 phpinfo();는 초기에 테스트 용도로만 사용해야 합니다. 잘못된 사용으로 인해 외부 사용자로부터 자신의 서버 정보가 공개될 수 있습니다.

PHP의 환경 정보는 php.ini 파일을 통해 변경할 수 있습니다.

**3.7 기호**

PHP와 더불어 많은 프로그램 언어들은 문법을 구성하고 구분을 하는 기호를 사용합니다. PHP 인터프리터는 C 언어로 개발되었고, 또한 문법적 표현과 기호들도 C 언어와 유사한 기호들을 사용합니다.

PHP 언어를 작성하기 위해서 사용하는 몇 가지 기호들에 대해서 먼저 살펴보도록 하겠습니다

**3.7.1 소괄호 ()**

소괄호 ()는 대부분 값을 전달하는 경우에 많이 사용됩니다. 조건문의 판단 값을 입력할 때와 함수호출을 위한 인자 값을 전달할 때 자주 접할 것입니다. 또한 소괄호는 일반적인 연산의 우선순위나 그룹을 지정할 때도 사용합니다.

소괄호는 항상 쌍으로 사용을 합니다. 만일 한 개, 두 개 등 다수의 값을 전달할 때도 좌우로 소괄호로 둘러싸인 쌍의 모습을 보입니다.

**3.7.2 중괄호**

중괄호는 {}는 문법적인 코드의 그룹을 지정할 때 많이 사용합니다. 여러 코드들을 묶어서 처리할 때 많이 사용합니다.

프로그램은 하나의 문장씩 구분하여 프로그램을 실행합니다. 하지만 조건문, 반복문, 함수, 클래스 등 여러 줄의 문장을 하나의 그룹으로 지정하고자 할 때 위아래로 중괄호를 자주 접할 수 있습니다.

중괄호는 반드시 {} 한 쌍으로 구성되어야 합니다. 만일 한 쌍으로 끝나지 않고 한쪽이 누락된다면 프로그램은 오류를 발생합니다.

중괄호는 계층화하여 작업이 가능합니다. 즉, 하나의 중괄호 안에 또 다른 문장과 중괄호를 입력할 수 있습니다. 이런 경우 중괄호의 위치와 코드를 이해하기 쉽게 하기 위해서 들여쓰기를 같이 사용합니다.

**3.7.3 대괄호**

대괄호 []는 배열과 선택할 값의 인자로 전달할 때 주로 사용합니다. 보통 변수명 뒤에 붙여서 사용합니다.

대괄호 안에 값을 선택하고자 하는 정수 값 또는 키 값을 넣어서 선택할 수 있습니다.

$arr[1];

$arr[‘name’];

­­또는 변수에 키 값 또는 정수 값을 넣어서 사용할 수도 있습니다.

$key = “name”;

$arr[$key];