اولین قانون در PHP

اولین و درعین‌حال، ساده‌ترین قانون PHP که مهم‌ترین قانون نیز می‌باشد، آن‌است که هر دستور، باید با سمی‌کالِن (;) خاتمه‌یابد. برای مثال، دستور زیر موجب بروز خطا خواهد شد:

```
echo("Hello")
```

برای اصلاح آن، باید این‌گونه بنویسید:

```
echo("Hello");
```

توضیحات در PHP

همان‌طور که احتمالاً می‌دانید، PHP یک زبان برنامه‌نویسی است. ساختار این زبان بسیار شبیه به زبان C++ است. یکی‌از امکانات سودمند در هر زبان برنامه‌نویسی، توضیحات هستند که برای کسی که بعداً قصد مطالعه کد را دارد، بسیار مفید است. در PHP برای درج توضیحات تک‌سطری در برنامه می‌توان از علامت‌های # و // استفاده‌نمود. برای مثال، عبارات زیر، هر دو، توضیح هستند و پردازش نمی‌شوند:

```
#This is a comment
//This is also a comment
```

برای توضیحات چندسطری نیز می‌توانید ابتدای توضیحات را با /* و پایان آنرا با */ مشخص‌کنید:

```
/*This is
a multiline
comment*/
```

متغیرها

هر زمان صحبت از زبان‌های برنامه‌نویسی به‌میان می‌آید، بدون‌شک اولین مفهومی که مطرح می‌شود، متغیر است. متغیر مکانی از حافظه است که دارای نام می‌باشد و توسط نام، می‌توان به آن دسترسی‌داشته و درصورت لزوم،آنرا تغییر داد. در PHP متغیرها با استفاده از علامت $ (دلار) تعریف می‌شوند. مثلاً $myVariable یک متغیر به‌نام myVariable است. نام متغیرها در PHP نسبت‌به بزرگی و کوچکی حروف، حساس است. برای مثال، دو متغیر $var و $VAR در PHP با یکدیگر متفاوتند. برای نام‌گذاری متغیر در PHP باید از دو قانون زیر پیروی‌کنیم:

۱- نام متغیر فقط می‌تواند شامل حروف (A-Z و a-z)، اعداد (0-9) و کارکتر خط‌زیر یا Underline (_) باشد.

۲- نام متغیر نمی‌تواند با عدد شروع‌شود.

بنابراین ازبین اسامی زیر، فقط سه مورد اول صحیح است:

```
$name              ✓
$_Name20           ✓
$my_1st_Name       ✓
$2_name            ✘
```

در PHP نوع متغیر تعیین نمی‌شود. درعوض، یک متغیر می‌تواند هر نوع مقداری را بپذیرد و نوع آن، به‌طور خودکار براساس محتوای آن، تغییر خواهد کرد. مثال:

```
$var=5;            ✓
$var="PHP";        ✓
$var=3.14;         ✓
```

هر سه دستور فوق، صحیح است.

مقداردهی به‌روش مقداری و ارجاعی

به دستورات زیر دقت‌کنید:

```
$x=5;
$y=$x;
$x=6;
```

در این مثال، بعد از اجرای دستورات فوق، مقدار $y برابر با 5 خواهد بود؛ زیرا در زمان مقداردهی، یک کپی از مقدار $x درون متغیر $y قرار گرفته‌است و درصورت تغییر مقدار $x، مقدار $y بدون تغییر خواهد ماند.

حال به دستورات زیر توجه‌کنید:

```
$x=5;
$y=&$x;
$x=6;
```

در این مثال، به‌دلیل استفاده از کارکتر & قبل‌از متغیر $x، به‌جای قراردادن یک کپی از مقدار $x درون $y، یک ارجاع به $x درون $y قرار می‌گیرد و درنتیجه، $y به همان محلی از حافظه اشاره می‌کند که $x اشاره دارد. بنابراین، درحقیقت، $y یک اسم مستعار (Alias) برای $x خواهد شد. به چنین شرایطی، مقداردهی ارجاعی می‌گویند. درنتیجه بعد از اجرای دستورات فوق، $y نیز مقدار 6 را در خود خواهد داشت.

انواع متغیرها در PHP

به‌طور کلی، هر متغیر در PHP می‌تواند یکی از انواع داده‌ای زیر را بپذیرد:

```
Integer
Floating-Point
String
Object
Array
```

که به‌ترتیب، برای نگه‌داری اعداد صحیح، اعداد اعشاری، رشته‌ها، اشیاء و آرایه‌ها به‌کار می‌روند.

آشنایی با دستور echo

این دستور، هرچه که به‌عنوان پارامتر دریافت‌کند را در محل قرارگرفتن مکان‌نما، در فایل HTML خروجی (که کاربر ملاحظه خواهد کرد)، می‌نویسد. برای مثال، دستورات زیر را درنظر بگیرید:

```
<HTML>
<HEAD>
<TITLE>Simple PHP</TITLE>
</HEAD>
<BODY>
<?PHP
	$x=5;
	$y=6;
	$z=$x+$y;
	echo("$z\n");
?>
</BODY>
</HTML>
```

کدهای HTML که خارج‌از تگ PHP قراردارند، عیناً در فایل HTML خروجی نوشته می‌شوند؛ اما کدهای PHP ابتدا پردازش‌شده و سپس، هرآنچه دستور echo به خروجی منتقل‌کند، در خروجی نوشته می‌شوند. درنتیجه، درصورتی‌که کد فوق را در یک فایل با پسوند php. ذخیره‌کرده و آنرا در مرورگر مشاهده‌کنید، عدد 11 را در صفحه خواهید دید و اگر برروی صفحه کلیک راست کرده و گزینه View Source را برگزینید، کد زیر را ملاحظه خواهید کرد:

```
<HTML>
<HEAD>
<TITLE>Simple PHP</TITLE>
</HEAD>
<BODY>
11
</BODY>
</HTML>
```

در این کد، کلیه عبارات (تا ابتدای تگ PHP عیناً در کد خروجی ظاهر شده‌اند. سپس کدهای PHP پردازش شده و مجموع متغیر $x با مقدار 5 و متغیر $y با مقدار 6، مقدار متغیر $z یعنی 11 را ساخته‌است. درنتیجه، دستور echo مقدار این متغیر یعنی 11 را در کد خروجی، بعد از دستور بازکردن تگ BODY نوشته‌است. سپس تگ PHP بسته‌شده و مابقی عبارات که HTML هستند، عیناً در کد خروجی ظاهر شده‌اند.

کار با رشته‌ها

رشته‌ها، مجموعه‌ای از کارکترهای متوالی هستند که بین دو گیومه تک (') یا دو گیومه جفت (") قرار می‌گیرند. مثال:

```
$a='This is a text.';
$b="This is a text too.";
```

درصورتی‌که بخواهید کارکتر ' (گیومه تک) را در رشته‌هایی که بین دو گیومه تک قراردارند درج‌کنید، باید یک کارکتر \ قبل‌از آن قراردهید:

```
$text='It\'s mine!';
```

به‌همین‌ترتیب، برای درج کارکتر " (گیومه جفت) در رشته‌های محصورشده بین دو گیومه جفت، باید یک کارکتر \ قبل‌از آن بگذارید:

```
$text="My friend's name is \"Ali\" and he's a good person.";
```

به کارکتر \ اطلاحاً ESCAPE (فرار) می‌گویند؛ زیرا برای فرار از معنای اصلی کارکترهای بعد از آن (و اعطای مفهومی دیگر به آنها) به‌کار می‌رود. در جدول زیر، تعدادی از کدهای ESCAPE را ملاحظه می‌کنید:

کد ESCAPE	نتیجه
\n	حرکت به ابتدای سطر بعد
\r	حرکت به ابتدای سطر جاری
\t	کارکتر TAB (معادل 8 کارکتر SPACE)
\\	کارکتر \
\"	کارکتر " (در رشته‌های محصور بین یک جفت ")
\'	کارکتر ' (در رشته‌های محصور بین یک جفت ')
\$	کارکتر $
\[0-7]	کارکتری که کد ASCII آن در مبنای هشت در جلوی آن نوشته شده‌است
\x[0-F]	کارکتری که کد ASCII آن در مبنای شانزده در جلوی آن نوشته شده‌است

ادغام رشته‌ها

برای ادغام رشته‌ها در PHP از کارکتر نقطه (.) استفاده می‌شود. برای‌مثال:

```
$a="PHP";
$b="Programmer";
$c=$a." ".$b; //$c="PHP Programmer";
```

در PHP می‌توانید رشته‌ها را با اعداد نیز ترکیب‌کنید:

```
$num=5;
$x="Test".$num; //$x="Test5";
```

تفاوت رشته‌های محصور بین گیومه جفت و تک

دستورات زیر را درنظر بگیرید:

```
$x=5;
$text1="X is $x";
$text2='X is $x';
```

در نتیجه دستورات فوق، عبارت X is 5 در متغیر $text1 و عبارت X is $x در متغیر $text2 است. علت این تفاوت در عملکرد، آن‌است که اسامی متغیرها در رشته‌هایی که بین دو گیومه جفت قراردارند، پردازش شده و به‌جای نام آنها، از مقدارشان در عبارت استفاده می‌شود؛ حال‌آنکه در رشته‌های محصور بین دو گیومه تک، هیچ پردازشی روی رشته انجام نمی‌شود و به‌همان‌شکل که نوشته می‌شود، مورد استفاده قرار خواهد گرفت. البته این پردازش، فقط مختص متغیرها است و هیچ‌گونه فراخوانی تابع (در آینده توضیح داده خواهد شد) یا عمل محاسباتی ریاضی پردازش نخواهد شد. برای مثال، نتیجه دستورات زیر:

```
$x=5;
$text1="X=($x+5)";
```

ذخیره عبارت X is (5+5) درون متغیر $text1 است. برای محاسبه صحیح مجموع و درج آن در رشته، باید به‌صورت زیر عمل‌شود:

```
$text1="X=".($x+5);
```

درنتیجه، در انتهای عبارت X=، نتیجه محاسبه ($x+5) به‌صورت یک رشته، ادغام‌شده و عبارت حاصل، درون متغیر $text1 قرار می‌گیرد.

استفاده از رشته به‌عنوان عدد

همان‌طور که در مثال قبل ملاحظه‌کردید، تبدیل عدد به رشته، در زمان نیاز، به‌طور خودکار انجام می‌شود. عکس این‌موضوع نیز صحیح‌است و رشته‌ها نیز درصورتی‌که در عبارات محاسباتی ریاضی مورداستفاده قرارگیرند، به‌عدد تبدیل می‌شوند؛ بدین‌ترتیب که از ابتدای رشته، تا جایی که با یک کارکتر غیرعددی برخورد نشود، جداشده و به‌صورت عدد تعبیر خواهد شد. اگر در ابتدای رشته، عددی وجود نداشته‌باشد، یک ثابت خاص به‌نام NaN (مخفف Not a Number) بازگردانده خواهد شد. مثال:

```
$text="52Ali";
$y=7+$text;        //$y=59;
```

محاسبه طول رشته

توسط تابع strlen می‌توان طول یک رشته را محاسبه‌کرد:

```
$text="Alireza";
$len=strlen($text); //$len=7;
```

عملگرها در PHP

تاکنون با روش تعریف و مقداردهی متغیرها آشنا شدیم. حال قصد داریم روش تغییر و دستکاری آنها را بیان‌کنیم. برای این‌کار، از عملگرها استفاده می‌شود. ساده‌ترین عملگر در PHP، عملگر انتساب است که با نماد = بیان می‌شود. برای مثال، دستور:

```
$x=5;
```

مقدار 5 را به متغیر $x نسبت می‌دهد؛ یا دستور:

```
$y=$x;
```

مقدار متغیر $x را به متغیر $y منسوب می‌کند. از این عملگر برای مقداردهی رشته‌ها نیز می‌توان استفاده‌نمود:

```
$t="Hello!";
```

عملگرهای ریاضی

PHP از پنج‌نوع عملگر ریاضی پشتیبانی می‌کند:

نام عملگر	نماد	مثال	توضیح
جمع	+	$a+$b	جمع $a و $b
تفریق	-	$a-$b	تفریق $b از $a
ضرب	*	$a*$b	ضرب $a در $b
تقسیم	/	$a/$b	تقسیم $a بر $b
باقیمانده	%	$a%$b	باقیمانده تقسیم $a بر $b

برای مثال، درصورتی‌که بخواهیم حاصل‌ضرب متغیرهای $x و $y را در $z ذخیره‌کنیم، باید این‌گونه بنویسیم:

```
$z=$x*$y;
```

هم‌چنین اگر بخواهیم به مقدار قبلی متغیر $a، 5 واحد اضافه‌کنیم، از این دستور استفاده می‌کنیم:

```
$a=$a+5;
```

البته در چنین حالت‌هایی که متغیر سمت‌چپ تساوی، بلافاصله بعد از عملگر انتساب ظاهر می‌شود، ساختار بهینه‌تری وجود دارد:

```
$a+=5;
```

دقت‌کنید که اگر بلافاصله بعد از عملگر انتساب، متغیر سمت چپ تساوی وجود نداشته‌باشد، نمی‌توان از این ساختار استفاده‌کرد:

```
$b=5-$b;      ✓
$b-=5;        ✘      //Result: $b=$b-5; not $b=5-$b;
```

به‌علاوه، برای افزایش و کاهش به‌میزان یک‌واحد نیز روشی باز هم ساده‌تر وجود دارد:

```
$a=$a+1;          $b=$b-1;
$a+=1;            $b-=1;
$a++;             $b--;
++$a;             --$b;
```

نتیجه اجرای تمامی دستورات دو گروه فوق، یکسان‌است. به دو دستور آخر هر گروه دقت‌کنید. درنتیجه اجرای هرکدام از دو دستور آخر گروه اول، مقدار متغیر $a یک‌واحد افزایش‌یافته وبا اجرای هرکدام از دو دستور آخر گروه دوم، مقدار متغیر $b یک‌واحد کاهش می‌یابد. تفاوت این دو دستور، در زمان افزایش یا کاهش است. برای درک بهتر، به مثال زیر دقت‌کنید:

```
$a=5;
$x="A=".$a++;
```

در این دستور، چون ابتدا $a ذکرشده و سپس، عملگر افزایش یک‌واحدی مورد استفاده قرار گرفته‌است، ابتدا مقدار فعلی آن (یعنی 5) در رشته $x قرارگرفته و سپس، یک‌واحد به آن افزوده می‌شود؛ لذا رشته $x حاوی عبارت A=5 خواهد بود. حال اگر همان‌دستور را به‌صورت زیر بنویسیم:

```
$x="A=".++$a;
```

ابتدا به متغیر $a یک‌واحد افزوده‌شده و سپس، در عبارت مورد استفاده قرار می‌گیرد؛ لذا رشته $x حاوی عبارت A=6 خواهد بود.

عملگرهای مقایسه‌ای

از این عملگرها برای مقایسه دو عبارت به‌صورت ریاضی استفاده می‌شود.

عملگر	مثال	توضیح
==	$a==$b	بررسی تساوی مقدار (5 و 5.0 و "5" و... با هم برابرند)
===	$a===$b	بررسی تساوی مقدار و نوع (5 و 5 با هم برابر و با 5.0 و "5" و... متفاوتند)
!=	$a!=$b	بررسی عدم‌تساوی مقدار (5 و 6 با هم متفاوتند ولی 5 و 5.0 و "5" و... با هم برابرند)
!==	$a!==$b	بررسی عدم‌تساوی مقدار و نوع (5 و 6 و 5.0 و "5" و... با هم متفاوتند)
>	$a>$b	بررسی بزرگ‌تربودن
<	$a<$b	بررسی کوچک‌تربودن
>=	$a>=$b	بررسی بزرگ‌تر یا مساوی بودن
<=	$a<=$b	بررسی بزرگ‌تر یا مساوی بودن

عملگرهای منطقی

از این عملگرها برای ترکیب شرایط مختلف جهت بررسی‌کردن استفاده می‌شود:

عملگر	مثال	توضیح
and	$a<5 and $b>2	هر دو شرط باید برقرار باشند
&&	$a<5 && $b>2	هر دو شرط باید برقرار باشند
or	$a<5 or $b>2	کافی‌است یکی‌از دو شرط برقرار باشند
||	$a<5 || $b>2	کافی‌است یکی‌از دو شرط برقرار باشند
xor	$a<5 xor $b>2	فقط باید یکی‌از دو شرط برقرار باشد
^	$a<5 ^ $b>2	فقط باید یکی‌از دو شرط برقرار باشد
!	!$a<5	شرط نباید برقرار باشد

اولویت عملگرها

درصورتی‌که در یک عبارت، از چند عملگر استفاده‌کنید، نتیجه عبارت با توجه به این‌که کدام‌عملگر ابتدا ارزیابی‌شود، متفاوت خواهد بود. برای مثال، در عبارت 2*4+5، اگر ابتدا + عمل‌کند، نتیجه، 18 و درصورتی‌که ابتدا * عمل‌کند، نتیجه 13 خواهد بود. در چنین شرایطی، اولویت عملگرها مشخص‌کننده ترتیب ارزیابی آنهاست. در PHP عملگرها به‌ترتیب زیر ارزیابی می‌شوند (عملگرهای سطرهای بالاتر، اولویت بیشتری دارند و عملگرهای یک‌سطر، دارای اولویت برابر هستند و به همان‌ترتیب نوشته‌شدن در عبارت، ارزیابی می‌شوند:

```
[       (
!       ~       ++      --
*       /       %
+       -       .
<       <=      >       >=
==      !=      ===     !==
&
^
|
&&      and
xor
||      or
?:
=       +=      -=      *=      /=      %=      .=      &=      |=      ^=      ~=
```

کنترل روند اجرای برنامه

تمامی برنامه‌هایی که تا اینجا بررسی‌کردیم، بدین‌صورت بودند که از ابتدا تا انتها، خط‌به‌خط اجرا می‌شدند. با استفاده از ساختارهای کنترلی، می‌توانیم این روش اجرا را تغییر دهیم.

ساختار if

این ساختار برای کنترل یک یا چند شرط و اجرای یک یا چند دستور درصورت برقراری یا عدم‌برقراری آن شرط یا شرایط، به‌کار می‌رود. ساختار کلی این دستور به‌صورت زیر است:

```
if( CONDITION(S)1 )
{
      true 1 block
}
elseif( CONDITION(S)2 )
{
      true 2 block
}
...
elseif( CONDITIONS(S)n )
{
      true n block
}
else
{
      false block
}
```

توضیح: درصورتی که شرط یا شرایط 1 برقرار باشند، بلاک true 1 اجرا می‌شود. درغیر این‌صورت، شرط یا شرایط 2 بررسی می‌شوند و درصورت برقرار بودن آنها، بلاک true 2 اجرا می‌گردد. درصورت عدم برقراری این شرط یا شروط نیز شرط یا شروط بعدی تا n (هر اندازه که باشند)، بررسی می‌گردند و درصورت برقراری هرکدام از شرط‌ها، بلاک کد مربوطبه همان شرط اجرا می‌شود. درنهایت، اگر هیچ‌کدام از بلاک‌های if یا elseif برقرار نبودند، بلاک false (کد مربوطبه else) اجرا می‌شود. در ساختار فوق، تمامی قسمت‌های elseif و قسمت else، اختیاری هستند و بنا به نظر برنامه‌نویس، می‌توان از آنها استفاده‌نمود. ضمناً اگر در هرکدام از بلاک‌ها، فقط یک دستور وجود داشته‌باشد، می‌توان از { و } صرف‌نظر نمود؛ هرچند به‌دلیل کاهش خوانایی برنامه و خطر بروز اشکالات احتمالی، این‌کار پیشنهاد نمی‌شود. برای درک بهتر، به مثال زیر دقت‌کنید:

```
if($day==0)
{
      $wday="Saturday";
}
elseif($day==1)
{
      $wday="Sunday";
}
...
$elseif($day==7)
{
      $wday="Friday";
}
else
{
      $wday="ERROR";
}
```

در مثال فوق، مقدار متغیر $day به‌ترتیب با مقادیر 0 تا 7 مقایسه می‌شود و درصورت برابربودن با هرکدام از آنها، نام روز مربوطه در متغیر $wday قرار می‌گیرد. درصورتی‌که مقدار متغیر $day هیچ‌کدام از مقادیر فوق نباشد، عبارت ERROR درون متغیر $wday قرار خواهد گرفت.

ساختار switch

اگر به‌خوبی به ساختار if مثال قبل دقت‌کنید، خواهید دید که در آن، مقدار متغیر $day با مقادیر مختلف بررسی‌شده و درصورت برابربودن با هرکدام از مقادیر، یک بلاک کد اجرا می‌شود. در چنین مواردی (ارزیابی یک متغیر با مقادیر مختلف و انجام کارهای متفاوت براساس مقادیر مختلف آن)، ساختار بهینه‌تری وجود دارد:

```
switch (VARIABLE)
{
	case VALUE 1:
		true 1 block
		break;
	case VALUE 2:
		true 2 block
		break;
	...
	case VALUE n:
		true n block
		break;
	default:
		false block
		break;
}
```

در این ساختار، مقدار VARIABLE به‌صورت بسیار سریع با تمامی caseها مقایسه می‌شود و درصورت برابربودن با هرکدام از آنها، بلاک کد همان‌قسمت تا زمان رسیدن به اولین دستور break; اجرا می‌شود. درصورتی‌که مقدار متغیر با هیچ‌کدام از مقادیر caseها برابر نباشد، بلاک کد مربوطبه قسمت default اجرا خواهد شد. برای درک بهتر، مثال قبل را با این ساختار بازنویسی می‌کنیم:

```
switch($day)
{
	case 0:
		$wday="Saturday";
		break;
	case 1:
		$wday="Sunday";
		break;
	...
	case 7:
		$wday="Friday";
		break;
	default:
		$wday="ERROR";
		break;
}
```

درمورد این ساختار دقت‌کنید که همیشه نمی‌توان از آن به‌عنوان جایگزین if استفاده‌نمود. برای استفاده از این ساختار، باید شرایط زیر فراهم‌باشد:

۱- اجرای بلاک‌های مختلف، بستگی به مقادیر مختلف یک متغیر داشته‌باشد (اگر بخواهیم چند شرط یا چند متغیر را بررسی‌کنیم، باید از if استفاده‌کنیم).

۲- مقادیر کاملاً مشخص و متمایز باشند (برای مثال، اگر بخواهیم درصورت بزرگتربودن از یک مقدار، کارهایی انجام‌دهیم، باید از if استفاده‌شود).

ضمناً در پایان هر case، باید از break; استفاده‌شود؛ درغیر این‌صورت، دستورات آن case تا زمان رسیدن به اولین break; یا رسیدن به انتهای ساختار switch، اجرا خواهند شد. البته این مسئله، در برخی‌موارد، سودمند است. مثال:

```
switch($operator)
{
	case '+':
		$result=$x+$y;
		break;
	case '-':
		$result=$x-$y;
		break;
	case '/':
		if($y!=0)
		{
			$result=$x/$y;
		}
		else
		{
			$result="DIVISION BY ZERO";
		}
		break;
	case '*':
	case 'x':
	case 'X':
		$result=$x*$y;
		break;
	default:
		$result="INVALID OPERATOR";
		break;
}
```

در این مثال، درصورتی‌که $op برابر با یکی‌از کارکترهای *، x یا X باشد، حاصل‌ضرب $x و $y را در $result ذخیره می‌کند.

ساختار :?

معمولاً بسیاری از دستورات if با ساختاری مشابه مثال زیر مورد استفاده قرار می‌گیرند:

```
if($x%2==0)
{
	echo("X is even.<BR/>\n");
}
else
{
	echo("X is odd.<BR/>\n");
}
```

همان‌طور که ملاحظه می‌کنید، بخش اعظمی از کد بلاک‌های true و false مشابه است. بنابراین می‌توانیم از ساختار :? برای خلاصه‌کردن این ساختار بهره‌مند شویم. این ساختار، به‌صورت زیر است:

```
CONDITION(S)?TRUE BLOCK:FALSE BLOCK
```

برای درک بهتر، مثال قبل را با این ساختار بازنویسی می‌کنیم:

```
echo("X is " . ($x%2==0?"even":"odd") . ".<BR/>\n");
```

ساختار while

از این ساختار برای تکرار بخشی از دستورات تا زمانی که یک شرط برقرار است، استفاده می‌شود و ساختار آن، به‌صورت زیر است:

```
while( CONDITION(s) )
{
      code block
}
```

در این ساختار، بلاک code تا زمانی که شرط یا شرایط دستور while برقرار باشند، تکرار می‌شود. بنابراین باید در بدنه بلاک مذکور، شرایطی درنظر گرفته‌شود که بالأخره در مرحله‌ای، شرط یا شرایط while نقض‌شود تا برنامه بتواند از حلقه تکرار، بیرون آید. درغیر این‌صورت، برنامه در یک حلقه بی‌نهایت گرفتار شده و نمی‌تواند کار خود را به‌درستی انجام‌دهد. مثال:

```
$i=1;
while($i<100)
{
      echo("Hello<BR/>\n");
      $i++;
}
```

با اجرای ساختار فوق، صدبار عبارت Hello در کد خروجی نوشته خواهد شد (و درنتیجه، به کاربر نشان‌داده می‌شود). اگر در ساختار قبل، دستور $i++; را ننویسیم، برنامه در حلقه گرفتار خواهد شد، زیرا $i همیشه 1 خواهد بود و همیشه، 1 از 100 کوچک‌تر است!

ساختار do...while

ساختار while، شرط خود را در ابتدا بررسی می‌کند و اگر در همان ابتدای کار، شرط برقرار نباشد، حتی یک‌بار هم حلقه اجرا نخواهد شد؛ اما گاهی‌اوقات لازم‌است که بلاک موردنظر، یک‌بار اجرا شود و سپس، شرط بررسی‌گردد و درصورت برقراربودن، مجدداً تکرار گردد. در چنین شرایطی، از ساختار do...while استفاده می‌کنیم:

```
do
{
      code block
}while( CONDITION(S) );
```

در این ساختار، شرط در پایان بررسی می‌شود.

ساختار for

این دستور، کامل‌ترین و درعین‌حال، پیچیده‌ترین ساختار تکرار در PHP است. در این ساختار، سه‌بخش در بدنه حلقه درنظر گرفته شده‌است: مقداردهی اولیه، شرط ادامه حلقه و افزایش/کاهش متغیر یا متغیرهای حلقه. ساختار کلی این دستور به‌صورت زیر است:

```
for( INITIALIZATION ; CONDITION(S) ; INCREMENT/DECREMENT )
{
      code block
}
```

روش اجرای این ساختار بدین‌صورت است که ابتدا قسمت INITIALIZATION اجرا می‌شود. سپس شرط یا شرایط قسمت CONDITIONS(S) بررسی‌شده و درصورت برقراربودن، code block اجرا می‌گردد. پس‌از پایان اجرای code block، قسمت INCREMENT/DECREMENT اجرا می‌گردد و مجدداً قسمت CONDITION(S) بررسی می‌گردد. درصورتی‌که هنوز شرایط برقرار باشند، حلقه مجدداً تکرار می‌گردد و این مراحل، تا زمانی که بالأخره، قسمت CONDITION(S) نقض‌شود، تکرار خواهد شد. برای درک بهتر، به همان مثال while که صدبار پیغام Hello را در کد خروجی می‌نوشت، دقت‌کنید که با ساختار for بازنویسی شده‌است:

```
for($i=1;$i<=100;$i++)
{
	echo("Hello<BR/>\n");
}
```

البته می‌توان هرکدام از قسمت‌های فوق را نادیده‌گرفت؛ مشروط بر آنکه در اجرای حلقه مشکلی پیش نیاید:

```
$i=1;
for(;$i<=100;)
{
	echo("Hello<BR/>\n");
	$i++;
}
```

البته این شکل استفاده از ساختار for، معمول نیست.

دستورات break; و continue; در حلقه‌های تکرار

کاربرد دستور break; در حلقه‌های تکرار بدین‌صورت است که هرگاه برنامه به دستور فوق برسد، بدون توجه به اتمام قانونی حلقه (با نقض‌شدن شرط تکرار حلقه)، از آن خارج می‌شود و برنامه را از اولین دستور بعد از حلقه، ادامه می‌دهد. دستور continue; نیز موجب می‌شود که PHP، تا انتهای بلاک کد را نادیده گرفته و به‌سراغ تکرار بعدی حلقه برود. مثالی از کاربرد دستور break; را در ادامه ملاحظه می‌کنید:

```
$i=1;
while(true)
{
	echo("$i<BR/>\n");
	$i++;
	if($i>100)
	{
		break;
	}
}
```

این حلقه (کمی عجیب)، اعداد از 1 تا 100 را در کد خروجی می‌نویسد. مثالی از کاربرد دستور continue; نیز به‌صورت زیر است:

```
for($i=1;$i<=100;$i++)
{
	if($i%2==0)
	{
		continue;
	}
	echo("$i<BR/>\n");
}
```

این حلقه نیز اعداد فرد از 1 تا 100 را در کد خروجی می‌نویسد. دقت‌کنید که چگونه درصورت زوج‌بودن $i، بقیه بدنه حلقه نادیده گرفته‌شده و به‌سراغ تکرار بعدی حلقه که یک $i فرد است، می‌رود.