הבסיס

התחביר הבסיסי

כללי התחביר של PHP פשוטים וקלים ללימוד. את שורות הקוד שכתובות ב-PHP ניתן לשלב בקבצי XML\HTML\XHTML (וכדומה) באמצעות תגיות PHP מיוחדות באופן הבא:

```
<?php  ... ?>
```

כך לדוגמא, הקובץ הבא כולל לצד תגיות HTML גם קוד ב-PHP אשר יתבצע בכל פעם שהשרת יקבל פניה להפעיל את הקובץ.

```
<b>PHP Simple Code Sample</b>
<?php
      $numA = 12;
      $numB = 4;
      $sum = $numA + $numB;
?>
The sum of <?php echo $numA." and ".$numB." is ".$sum ?>
```

לחילופין, ניתן להשתמש גם בתגיות הבאות:

תגיות מקוצרות

```
<? … ?>
```

תגיות כלליות לסימון סקריפט

```
<script language="php"> … </script>
```

תגיות בסגנון ASP

```
<% … %>
```

כל משפט ב-PHP צריך להסתיים ב ';'.

ניתן לשלב הערות במספר פורמטים שונים.

ניתן לשלב הערות בפורמט המקובל ב- C++:

```
// bla bla bla
```

ניתן לשלב הערות בפורמט המקובל ב- C:

```
/*    bla bla bla
      bla bla bla
      bla bla bla **/
```

ניתן לשלב הערות בפורמט המקובל ביצירת הערות API בתכניות שכתובות ב-Java:

```
/**
*       bla bla bla
*       bla bla bla
*/
```

ניתן לשלב הערות בפורמט ייחודי ל-PHP:

```
#       bla bla bla
```

בדומה לשפות תיכנות אחרות, שפת הפיתוח PHP איננה רגישה לריווחים. ניתן לשלב ריווחים ככל הנדרש כדי לקבל קוד מקור קריא וברור. לא תהיה לכך כל השפעה על אופן הריצה של הקוד.

בדומה לשפות תיכנות אחרות, ניתן לשלב בתכנית משפטים מורכבים. משפט מורכב הוא משפט אחד או יותר שתחומים בסוגריים מסולסלות.

```
{
        $numA = 4;
        $numB = 3;
        $sum = $numA + $numB;
}
```

הדפסה למסך

כדי להריץ להדפיס למסך (למסך הדפדפן) יש להשתמש בפקודה echo. זו איננה פונקציה.

```
<?php
echo "Hello World";
?>
```

ניתן לשרשר טקסטים באמצעות האופרטור נקודה '.' באופן הבא:

```
<?php
$strA = "Hello";
$strB = "World";
echo $strA." ".$strB;
?>
```

ניתן לשלב בתוך המחרוזת ששולחים להדפסה משתנים ובדרך זו לשלב את ערכם בטקסט שמודפס.

```
<?php
$num = 12;
echo "num=$num";
?>
```

טיפוסים

שפת התיכנות PHP תומכת בטיפוסים שונים של ערכים. את הטיפוסים השונים ניתן לסווג לשתי קטגוריות.

הקטגוריה הראשונה היא:
Compound Data Types

הקטגוריה השניה היא:
Scalar Data Types

הקטגוריה Scalar Data Types כוללת את האפשרויות הבאות:
boolean
int
float
string
כל ערך מטיפוס כל אחד מארבעת הטיפוסים הללו כולל ערך אחד בלבד. ערך מטיפוס boolean יכול להיות או 'true' או 'false'. ערך מטיפוס int יכול להיות ערך מספרי שלם בלבד. ערך מטיפוס int יכול להיות או ערך חיובי או ערך שלילי. ערך מטיפוס floar יכול להיות ערך מספרי ממשי. ערך מטיפוס float יכול להיות או ערך חיובי או ערך שלילי. ערך מטיפוס string יכול לכלול אוסף של תו אחד או יותר.

הקטגוריה Compound Data Types כוללת את האפשרויות הבאות:
Arrays
Objects
להבדיל מערכים מטיפוס אחת האפשרויות בקטגוריה Scalar Data Type ערך שהוא Array או Object יכול לכלול יותר מערך אחד. ערך שהוא Array מהווה אוסף מסודר של אלמנטים (לכל אלמנט יש מספר אינדקס). כל אלמנט יכול להיות מכל טיפוס. ערך שהוא Object מהווה אוסף של נתונים בצירוף קוד.

הערך המיוחד NULL ניתן להשמה אל תוך כל משתנה ומהווה אינדיקציה לכך שהמשתנה עדיין לא מכיל אף ערך. את הערך המיוחד NULL ניתן להשים אל תוך כל משתנה. לחילופין, אם יוצרים משתנה ולא משימים לתוכו אף ערך אז באופן אוטומאטי הוא יכיל NULL.

משתנה ב-PHP שמחזיק ב-reference לאובייקט אשר מייצג משאב חיצוני (לדוגמא: קובץ, תמונה, חיבור תקשורת וכו') נחשב למשתנה מסוג resource. משתנים מסוג resource נוצרים באמצעות פונקציות מיוחדות שמטרתן להחזיר reference לאובייקט אשר מייצג משאב חיצוני. פונקציות מיוחדות מאפשרות לנו להשתמש במשתנים השונים מסוג resource וכמו כן גם לשחרר את המשאבים החיצוניים שה-references אליהם מוחזקים בתוך אותם משתני resource שבהם אנחנו משתמשים.

ניתן להמיר את הטיפוס של ביטוי נתון לטיפוס אחר באמצעות ציון הטיפוס החדש שאליו רוצים להמיר בתוך סוגריים עגולות וכתיבתן לפני הביטוי שאת הטיפוס שלו רוצים להמיר.

```
<?php
$numA = 3.5;
$numB = 4.5;
$number = ((int)$numA)*((int)$numB);
echo "number=$number"
?>
```

משתנים

משתנה הוא מקום זמני בזיכרון אשר יש לו שם ושיכול להחזיק בערך מכל אחד מהטיפוסים האפשריים. שמו של משתנה ב-PHP תמיד יתחיל בתו $. שם של משתנה יכול לכלול תווים שהם אותיות, קו תחתי וספרות בלבד. התו השני (אחרי התו $) יכול להיות קו תחתי או אות בלבד.

```
<?php
$num = 2;
$_num = 3;
$num1 = 4;
// $8num = 5; # not legal!
?>
```

ערכו של משתנה יכול להיות שמו של משתנה אחר.

```
<?php
$varname = "var";
$$varname = 10;   // as if we refer to $var
echo $var;  // output 10
echo "<br>";
echo $$varname;  //output 10
?>
```

ערכו של משתנה יכול להיות שם של פונקציה. במקרה כזה, ציון שמו של המשתנה בתוספת סוגריים עגולות יגרות לקריאה להפעלת הפונקציה.

```
<?php
function doSomething() {echo 'Bonga Da'; }
$var = 'doSomething';
$var(); //that will result in calling the function
?>
```

באמצעות הפקודה `isset` ניתן לבדוק אם משתנה מסויים אכן קיים לפני שמנסים להשתמש בו. הפקודה מחזירה `true` במקרה שהמשתנה קיים ו-`false` במקרה שהמשתנה עדיין לא קיים.

```
<?php
if(isset($number))
{
      echo "number=$number";
}
else
{
```

```
	echo "The variable still doesn't exist...";
}
echo "<br>";
$number = null;
if(isset($number))
{
	echo "number=$number";
}
else
{
	echo "The variable still doesn't exist...";
}
echo "<br>";
$number = 4;
if(isset($number))
{
	echo "number=$number";
}
else
{
	echo "The variable still doesn't exist...";
}
?>
```

הפלט יהיה:

```
The variable still doesn't exist...
The variable still doesn't exist...
number=4
```

קבועים

קבוע ב-PHP הוא משתנה המכיל ערך מטיפוס boolean, int או float בלבד. בדומה למשתנה רגיל גם למשתנה שנחשב לקבוע יש רגישות לאותיות גדולות ו/או קטנות. להבדיל ממשתנה רגיל התו הראשון של משתנה שנחשב לקבוע לא יהיה התו '$'. נהוג ליצור קבועים ששמם כולל אותיות גדולות בלבד.

יוצרים קבוע באמצעות הפקודה define באופן הבא:

define (שמו של הקבוע, ערכו של הקבוע)

לדוגמא:

```
<?php
define('MAX_SPEED',120);
define('WEBSITE','www.zindell.com');
echo MAX_SPEED;
echo '<BR>';
echo WEBSITE;
?>
```

הפלט יהיה:

```
120
www.zindell.com
```

אופרטורים

שפת התיכנות PHP תומכת במגוון רחב של אופרטורים.

האופרטורים האריתמטיים הבינריים פועלים על שני אופרנדים ומאפשרים את ביצוע פעולות החשבון הבסיסיות:

+	פעולת חיבור
-	פעולת חיסור
*	פעולת כפל
/	פעולת חילוק
%	פעולת חישוב שארית מחלוקה

האופרטורים האריתמטיים האונרים פועלים על אופרנד אחד שחייב להיות משתנה ומבצעים פעולת הגדלה או הקטנה ביחידה אחת:

++	הגדלה ביחידה אחת
--	הקטנה ביחידה אחת

יש חשיבות למיקום של האופרטור האריתמטי האונרי לפני או אחרי שם של משתנה. אם ממקמים אותו לפני שם של משתנה אז ערכו של הביטוי כולו הוא הערך החדש של המשתנה. אם ממקמים אותו אחרי שם של משתנה אז ערכו של הביטוי הוא הערך הישן של המשתנה (לפני השינוי).

```
<?php
$numA = 2;
$numB = ++$numA;  // $numB will be assigned with 3
$numC = $numB--;  // $numC will be assigned with 3
echo "numA=$numA";  // output 3
echo "<BR>";
echo "numB=$numB";  // output 2
echo "<BR>";
echo "numC=$numC";  // output 3
?>
```

הפלט יהיה:

```
numA=3
numB=2
numC=3
```

האופרטור נקודה '.' מאפשר לשרשר מחרוזות תווים אחת לשניה.

```
<?php
$strA = "Hello";
$strB = "World";
```

```
echo $strA." ".$strB;
?>
```

האופרטורים שעובדים על ביטים כוללים את האופרטורים הבאים:

האופרטור & פועל על שני אופרנדים (ערכים מטיפוס `int`) ומחזיר ערך חדש מטיפוס `int` שכל אחד מהביטים שלו דלוק אם ורק אם הוא דלוק בכל אחד משני האופרנדים (הערכים מטיפוס `int`) שעליהם האופרטור פועל.

האופרטור | פועל על שני אופרנדים (ערכים מטיפוס `int`) ומחזיר ערך חדש מטיפוס `int` שכל אחד מהביטים שלו דלוק אם הוא דלוק בלפחות אחד משני האופרנדים (הערכים מטיפוס `int`) שעליהם האופרטור פועל.

האופרטור ^ פועל על שני אופרנדים (ערכים מטיפוס `int`) ומחזיר ערך חדש מטיפוס `int` שכל אחד מהביטים שלו דלוק אם הוא דלוק באופרנד אחד בלבד מבין שני האופרנדים (הערכים מטיפוס `int`) שעליהם האופרטור פועל.

האופרטור << פועל על שני אופרנדים (ערכים מטיפוס `int`) וגורם להזזה ימינה של הביטים באופרנד השמאלי במספר צעדים שזהה לערכו של האופרנד הימני.

האופרטור >> פועל על שני אופרנדים (ערכים מטיפוס `int`) וגורם להזזה שמאלה של הביטים באופרנד השמאלי במספר צעדים שזהה לערכו של האופרנד הימני.

האופרטור ~ פועל על אופרנד (ערך מטיפוס `int`) אחד בלבד וגורם לכך שכל ביט דלוק נכבה וכל ביט כבוי נדלק.

אופרטור ההשמה ב-`PHP` הוא הסימן '='. באמצעותו ניתן להכניס ערך לתוך משתנה. השמה ב-`PHP` נעשית `By Value` אלא אם הוספנו את הסימן '&' לפני שמו של המשתנה שאת ערכו אנחנו רוצים להכניס לתוך משתנה אחר.

```
<?php
$numA = 2;
$numB = $numA; // value of $numA is assigned By Value into $numB
$numC = &$numA; // value of $numA is assigned By Reference into $numC
echo "<BR>before...";
echo "<BR>".$numA;
echo "<BR>".$numB;
echo "<BR>".$numC;
$numA = 3;
echo "<BR>after...";
echo "<BR>".$numA;
echo "<BR>".$numB;
```

```
echo "<BR>".$numC;
?>
```

הפלט הוא:

```
before...
2
2
2
after...
3
2
3
```

האופרטורים לביצוע השוואה בין ערכים כוללים את האופרטורים הבאים:

האופרטור == מחזיר `true` אם שני האופרנדים שעליהם הוא פועל בעלי אותו ערך ומאותו טיפוס או שלחילופין הטיפוס שלהם יכול לעבור המרה באופן אוטומטי כך ששני האופרנדים יהיו מאותו טיפוס. המרה באופן אוטומטי נעשית בכל עת שיש בה צורך ושאין בה סכנה לפגיעה באחד הערכים.

האופרטור === מחזיר `true` אם שני האופרנדים שעליהם הוא פועל בעלי אותו ערך ושניהם מאותו טיפוס.

האופרטור =! מחזיר `true` אם שני האופרנדים שעליהם הוא פועל לא זהים בערכם. הטיפוס של כל אחד מהאופרנדים איננו חשוב כל עוד מתאפשרת המרה אוטומטית של אחד הטיפוסים כדי להשוותו לאחר.

האופרטור <> מחזיר `true` אם שני האופרנדים שעליהם הוא פועל לא זהים בערכם. הטיפוס של כל אחד מהאופרנדים איננו חשוב.

האופרטור ==! מחזיר `true` אם שני האופרנדים שעליהם הוא פועל לא זהים בערכם או שהם מטיפוס שונה.

האופרטור > מחזיר `true` אם האופרנד השמאלי קטן בערכו מהאופרנד הימני.

האופרטור => מחזיר `true` אם האופרנד השמאלי קטן או שווה בערכו לאופרנד הימני.

האופרטור < מחזיר `true` אם האופרנד השמאלי גדול בערכו מהאופרנד הימני.

האופרטור =< מחזיר `true` אם האופרנד השמאלי גדול או שווה בערכו לאופרנד הימני.

האופרטורים הלוגיים הבינריים פועלים על ערכים מטיפוס boolean. האופרטורים הלוגיים הבינריים כוללים את האופרטורים הבאים:

&& או AND
אופרטור זה מחזיר true אם ורק אם שני האופרנדים (הערכים הלוגיים) שעליהם הוא פועל הם true.

|| או OR
אופרטור זה מחזיר true אם לפחות אחד משני האופרנדים (הערכים הלוגיים) שעליהם הוא פועל הוא true.

XOR
אופרטור זה מחזיר true אם ורק אם רק אחד משני האופרנדים (הערכים הלוגיים) שעליהם הוא פועל הוא true.

האופרטור הלוגי האונרי פועל על ערך אחד מטיפוס boolean. ב-PHP קיים אופרטור לוגי אונרי אחד.

! או NOT
אופרטור זה מחזיר true אם האופרנד (הערך הלוגי) שעליו הוא פועל הוא false. אופרטור זה מחזיר false אם האופרנד (הערך הלוגי) שעליו הוא פועל הוא true.

האופרטור '@' ניתן להוספה לפני כל ביטוי שברצוננו לגרום לכך שבמידה שתתרחשנה שגיאות בעת הערכתו (ביצועו) אז הן מנוע ה-PHP יתעלם מהן (מרובן).

באמצעות האופרטור שכולל שתי מרכאות בודדות ('...') ניתן לגרום לביטוי שרושמים בינהן להתבצע כאילו כתבנו אותו בשורת הפקודה. במידה שיש פלט שמקבלים כתוצאה מהפעולה אז הוא יוחזר אל תוך המשתנה שלתוכו אנו מנסים להכניס את ערכו של הביטוי שרושמים. אופרטור זה לא עובד כאשר המצב safe mode מופעל או כאשר הפונקציה shell_exec היא במצב disabled.

סדר הקדימויות של האופרטורים מתואר בטבלה הבאה:

```
[
++ --
! ~ - (int) (float) (string) (array) (object) @
* / %
+ - .
<< >>
< <= > >=
== != === !==
&
^
|
&&
||
? :
= += -= *= /= .= %= &= |= ^== <<= >>=
and
xor
or
,
```

משפטי בקרה

שפת התיכנות PHP תומכת במרבית משפטי הבקרה שמוכרים משפות תיכנות אחרות.

משפט ה-if ומשפט ה-if..else

```
if(expression1)
{
	...
}
else
{
	...
}
```

אם הביטוי הלוגי `expression1` הוא `true` אז יתבצע המשפט המורכב (הבלוק) הראשון. אם הביטוי הלוגי `expression1` הוא `false` אז יתבצע המשפט המורכב (הבלוק) השני.

משפט האופרטור הטרנרי מאפשר מימוש של if..else בצורה מקוצרת בביטוי אחד.

```
$var = (expression1)?expression2:expression3;
```

אם ערכו של `expression1` הוא `true` אז אל תוך המשתנה `var` ייכנס הערך של `expression2` ואם ערכו של `expression1` הוא `false` אז אל תוך המשתנה `var` ייכנס הערך של `expression3`.

```
<?php
$num = (4>3)?12:14;
echo $num; //output 12
?>
```

משפט ה-`switch case` ב-`PHP` פועל בדומה למשפט ה-`switch case` ב-`Java` וב-`C` ו- `C++`.

```
switch($var)
{
	case ___:
	...
	break;
	case ___:
	...
	break;
	default:
	...
}
```

ערכו של המשתנה `var`$ מחושב ומושווה לכל אחד מה-`cases`. במידה שיש `case` שערכו זהה לערך של המשתנה `var`$ אז הביצוע עובר לפקודות של אותו `case`. במידה שאין `break` הביצוע ממשיך עד לסוף הבלוק או עד להגעה לפקודה `break`. אם

ערכו של המשתנה $var איננו זהה לאף אחד מה-cases אז הביצוע עובר לפקודות שמופיעות מייד לאחד default.

משפט הלולאה while פועל בדומה ל-C ו-Java:

```
while(expression1)
	statement
```

כל עוד ערכו של expression1 הוא true אז ה-statement מתבצע. ה-statement יכול להיות משפט פשוט או משפט מורכב (בלוק).

```
while(expression1)
{
	...
}
```

משפט הלולאה do while פועל בדומה ל-C ו-Java:

```
do
	statement
while(expression1)
```

המשפט statement מתבצע פעם אחת לפחות (להבדיל מלולאת while שבה המשפט statement עשוי גם בכלל לא להתבצע). המשפט statement מתבצע כל עוד ערכו של expression1 הוא true. ה-statement יכול להיות משפט פשוט או משפט מורכב (בלוק).

```
do
{
	…
}
while(expression1)
```

משפט ה-for ב-PHP דומה בפעולתו למשפט ה-for ב-Java, ב-C וב-C++:

```
for(expression1; expression2; expression3)
	statement
```

תחילה מתבצע expression1 אשר תפקידו בדרך כלל לאתחל את המשתנה אשר משמש את הלולאה. לאחר מכן, מחושב ערכו של expression2 אשר חייב להיות מטיפוס בוליאני. אם ערכו false אז המשפט (ה-statement) של הלולאה לא יתבצע בכלל. אם ערכו true אז המשפט (ה-statement) של הלולאה יתבצע ומייד לאחר מכן יתבצע expression3 שתפקידו לקדם את המשתנה שמשמש את הלולאה. מייד לאחר שמתבצע expression3 מתבצע חישוב של expression2 ובהתאם לערכו שוב יתבצע ה-statement (אם true) או שנצא מהלולאה (אם false)... וחוזר חלילה. בדומה למשפטי הבקרה האחרים גם בלולאה for ניתן להפעילה על משפט מורכב במקום משפט פשוט.

```
for(expression1; expression2; expression3)
{
      ...
}
```

להלן דוגמא ללולאת `for` אשר מדפיסה את כל המספרים בין `1` ו-`100`:

```
<?php
for($index=1; $index<=100; $index++)
{
      echo '<BR>'.$index;
}
?>
```

באמצעות הפקודה `break` ניתן להורות על יציאה מלולאה:

```
for(expression_1; boolean_exp; expression_2)
{
      ...
      ...
      if(...) break;
}
```

לפקודה `break` ניתן להוסיף מספר כדי להורות על הרמה שממנה יש לצאת (כאשר יש משפט בקרה אחד בתוך אחר וכו'... ניתן באמצעות המספר הנוסף להורות על הרמה שממנה יש לצאת):

```
for(exp_1; boolean_exp; exp_2)
{
      for(exp_1; boolean_exp; exp_2)
      {
            ...
            if(...) break 2; //exit both loops
      }
}
```

באמצעות הפקודה `continue` ניתן להורות על הפסקת האיטרציה הנוכחית ומעבר לאיטרציה הבאה:

```
for(exp_1; boolean_exp; exp_2)
{
      ...
      ...
      if(...) continue;
      ...
}
```

כאשר הפקודה `continue` מתבצעת אז הפקודות שמופיעות אחריה לא מתבצעות והביצוע עובר לביטוי `exp_2`, הביטוי השלישי של לולאת ה-`for`.