



Full Stack- server- PHP

הרצאה 4

משפטי תנאי

לולאות

קבצים



בקרת זרימה



תנאים-if/else/elseif

- if (*condition*) {
 code to be executed if condition is true;
}
- if (*condition*) {
 code to be executed if condition is true;
} else {
 code to be executed if condition is false;
}
- if (*condition*) {
 code to be executed if this condition is true;
} elseif (*condition*) {
 code to be executed if first condition is false and this condition is true;
} else {
 code to be executed if all conditions are false;
}

משפטי תנאי

- משפט תנאי מורכב מתנאי וממשהו שיתבצע במידה והתנאי הזה נכון.
- יש גם משפטי תנאי שבהם יש פירוט מה יתבצע אם התנאי הזה לא נכון.
- דוגמא:

```
<?php
    $my_var = 5;
    if ($my_var < 6)
    {
        print 'my_var is smaller than 6';
    }
    else
    {
        print 'my_var is bigger than 6';
    }
?>
```

■ יודפס:

my_var is smaller than 6



משפטי תנאי

```
<?php
    $my_var = 6;
    if ($my_var < 6)
    {
        print 'my_var is smaller than 6';
    }
    else if($my_var > 6)
    {
        print 'my_var is bigger than 6';
    }
    else
    {
        print 'my_var is equal to 6';
    }
?>
```

יודפס:

my_var is equal to 6

משפטי תנאי – שילוב של כמה תנאים

לפעמים קורה שצריך לשלב כמה תנאים. החיבור בין שני תנאים נעשה ע"י:

■ && עבור קשר "וגם"

■ || עבור קשר "או"

■ לדוגמא:

```
<?php
    $my_var = 5;
    if($my_var > 2 && $my_var < 4)
    {
        print 'my_var is between 2 and 4';
    }
    else
    {
        print 'my_var is NOT between 2 and 4';
    }
?>
```

■ יודפס
my_var is NOT between 2 and 4



תנאים - תרגיל

■ מה יהיה הפלט?

```
<?php
$my_var = 6;
if ($my_var < 0 || $my_var > 10 && $my_var %2 == 0)
{
    print 'my_var is > 2 or < 0 AND is even';
}
else
{
    print 'my_var is between 0 and 10 OR is odd';
}
?>
```

my_var is between 0 and 10 OR is odd-יודפס

שייוין

יותר טוב להסביר עם דוגמא, למשל: ➡

```
<?php
$my_var = 6;
if ($my_var ==6)
{
    print 'my_var is 6';
}
else
{
    print 'my_var is not 6';
}
?>
```

הפלט:
my_var is 6

■ אבל מה יקרה אם כותבים את הקוד הבא?

➡ האם לפי דעתכם PHP תמיר את המחרוזת למספר?

```
<?php
$my_var = '6';
if ($my_var ==6)
{
    print 'my_var is 6';
}
else
{
    print 'my_var is not 6';
}
?>
```

➡ אלו מכם שמכירים כמה שפות תכנות, יהמרו שלא, כי מספר לא יכול להיות שווה למחרוזת טקסט! אבל למרבה הצער, התשובה היא כן, והתנאי יתקיים ועל המסך יודפס **my_var is 6** למרות שב- **my_var** הכנסנו מחרוזת טקסט ולא מספר.



אי שיויון !

```
<?php
$my_var = '6';
if ($my_var !=6)
{
    print 'my_var is not 6';
}
else
{
    print 'my_var IS 6';
}
?>
```

הפלט:
my_var IS 6



סימן השיוויון עם < או >

```
<?php
$my_var = '6';
if ($my_var <=6)
{
    print 'my_var is smaller than 6 or equal to 6';
}
else
{
    print 'my_var is bigger than 6';
}
?>
```

הפלט:

my_var is smaller than 6 or equal to 6



משפט תנאי מקוצר

■ ניתן לקצר משפטי תנאי למשל:

```
<?php
    $my_var = 5;
    if($my_var == 5) {
        print "my_var is 5";
    }
    else {
        print "my_var is NOT 5";
    }
?>
```

■ בכתיב מקוצר יראה:

```
<?php
    $my_var = 5;
    $my_var == 5 ? print "my_var is 5" : print "my_var is NOT 5";
?>
```



משפט תנאי מקוצר - המשך

- כפי שניתן לראות, הוא הרבה יותר קצר והמבנה שלו מאד ברור:
 - בתחילה נכתב התנאי (בלי סוגריים) ולאחריו סימן שאלה
 - לאחר מכן נכתב מה יקרה אם התנאי יתמלא
 - נקודתיים
 - ובסוף מה יקרה אם התנאי לא מתמלא (אם אין דבר כזה אפשר לכתוב null או "")
- לדוגמא: האם התנאי נכון?

```
<?php
$my_var = 5;
$my_var < 6 || $my_var > 2 ? print "my_var between 2 and 6"
: print "my_var is NOT between";
?>
```

הפלט: ➡

my_var between 2 and 6 ➡



Switch-Case

```
<?php
    $my_var = 5;
    switch($my_var) {
        case 1:
            print "Sunday";
            break;

        case 2:
            print "Monday";
            break;

        case 3:
            print "Tuesday";
            break;

        <....>

        default:
            print "my_var is not 1-7!";
            break;
    }
```

- במידה ורוצים לכתוב סקריפט קטן שמחליף מספר ביום בשבוע - אם המשתנה הוא 'אחד' הסקריפט ידפיס את המילה 'Sunday', אם המשתנה הוא שתיים הסקריפט ידפיס את המילה 'Monday' וכך הלאה.

- ניתן להשתמש בשיטה ה- if וה- else אבל הקוד יהיה ארוך.

- במקום זה, ניתן להעמיד 'למבחן' את המשתנה ולציין מה יקרה כאשר המספר שלו שווה לאחת, שתיים, שלוש וכך הלאה.

- ניתן אפילו יכול לציין איזשהו default במידה ויש לו מספר שלא תואם לציפיות



משפטי switch

```
switch (n) {  
    case label1:  
        code to be executed if n=label1;  
        break;  
    case label2:  
        code to be executed if n=label2;  
        break;  
    case label3:  
        code to be executed if n=label3;  
        break;  
    ...  
    default:  
        code to be executed if n is different from all labels;  
}
```



switch-ל אמצע

```
<?php
$favcolor = "red";
switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```


- While
- Do.. While
- For
- foreach



לולאת while

`while (condition is true) {
 code to be executed;
}`

```
<?php  
$x = 1;  
  
while($x <= 5) {  
    echo "The number is: $x <br>";  
    $x++;  
}  
?>
```

The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5



לולאות do while

כשנרצה שהלולאה תרוץ לפחות פעם אחת

```
do {  
    code to be executed;  
} while (condition is true);
```

```
<?php
```

```
$x = 1;
```

```
do {
```

```
    echo "The number is: $x <br>";
```

```
    $x++;
```

```
} while ($x <= 5);
```

```
?>
```

The number is: 1

The number is: 2

The number is: 3

The number is: 4

The number is: 5



לולאת do...while

■ לולאת do while מתחילה ב-do ולאחריה סוגריים מסולסלות ולבסוף While עם סוגריים

■ בתוך הסוגריים המסולסלות יוגדרו הפקודות לביצוע

```
do
{

}
while( /* while condition */)
```

■ לולאת do...while הינה לולאת בקרת יציאה
■ הפקודות לביצוע יתרחשו לפחות פעם אחת



לולאת do...while - תרגיל

■ מה יהיה הפלט?

■ מה יהיה הערך של i בתום הרצה?

```
<?php
    $i = 4;
    do
    {
        print($i * 5 / 2);
        $i += 2;
    }
    while( $i<3);
?>
```

הפלט: 10 ■



הפקודה break

■ באמצעות הפקודה ניתן להורות על יציאה מלולאה

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

■ ניתן להוסיף מספר כדי 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
    }
}
```



הפקודה Continue

■ באמצעות הפקודה ניתן להורות על הפסקת האיטרציה הנוכחית ומעבר לאיטרציה הבאה

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

■ כאשר הפקודה מתבצעת אז הפקודות שמופיעות אחריה לא מתבצעות והביצוע עובר לביטוי exp_2 (הביטוי השלישי)



לולאות For

- לולאת for מתחילה ב- for עם סוגריים:

```
for( /* for statement */ )
```

- בתוך הסוגריים יש את ההצהרה של ה- for שזה לב הלולאה. היא תמיד מורכבת משלושה חלקים.

```
For( $i=0 )
```

- ראשית, הגדרת האינדקס:

- האינדקס הוא משתנה שרץ בכל פעם שהלולאה עושה סיבוב

- אנחנו יכולים לגרום לו שיתחיל מאפס או מכל מספר אחר

- שנית, הגדרת זמן ריצה:

```
for( $i=0; $i<6 )
```

- במקרה הזה, אנו מגדירים לו את התנאי שכל עוד הוא מתקיים, הלולאה תמשיך לרוץ.

- ברגע שהתנאי לא מתקיים, הלולאה מפסיקה להתקיים.



לולאות For

■ מבנה For - המשך

■ שלישית, הגדרת הצעד:

```
for( $i=0; $i<6; $i++ )
```

■ פה אנחנו אומרים ל-\$i כמה להתקדם בכל סיבוב

■ בעצם, הלולאה לוקחת משתנה ומשנה אותו
בהתאם לכללים שקבענו בתוך הסוגריים

■ בכל 'סיבוב' של המשתנה, מה שנמצא בתוך
הסוגריים המסולסלות מורץ

■ דוגמא:

```
<?php
    for($i = 0; $i < 6; $i++)
    {
        print "hello";
    }
?>
```



לולאת for

*for (init counter; test counter; increment counter) {
 code to be executed for each iteration;
}*

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10



לולאת foreach

```
foreach ($array as $value) {  
    code to be executed;  
}
```

```
<?php  
$colors = array("red", "green", "blue", "yellow");  
  
foreach ($colors as $value) {  
    echo "$value <br>";  
}  
?>
```

red
green
blue
yellow



שימוש ב- break ו- continue

■ Break משמש גם ליציאה מלולאה

```
<?php
for ($x = 0; $x < 10; $x++) {
    if ($x == 4) {
        break;
    }
    echo "The number is: $x <br>";
}
?>
```

The number is: 0
The number is: 1
The number is: 2
The number is: 3

■ Continue - מדלג על איטרציה אחת בלולאה

```
<?php
for ($x = 1; $x < 6;
    $x++) {
    if ($x == 4) {
        continue;
    }
    echo "The number is: $x
    <br>";
}
?>
```

The number is: 1
The number is: 2
The number is: 3
The number is: 5



פונקציות



סוגי פונקציות

- Built-in Functions - פונקציות מובנות
- User Defined Functions - פונקציות שנוצרות ע"י המשתמש

- ```
function functionName() {
 code to be executed;
}
```

שם הפונקציה חייב להתחיל באות או ב- \_  
אין משמעות לאותיות גדולות/קטנות



# קריאה לפונקציה

---

```
<?php
function writeMsg() {
 echo "Hello world!";
}
```

```
writeMsg(); // call the function
?>
```

פלט:

Hello world!



# שליחת משתנה לפונקציה

---

```
<?php
function familyName($fname) {
 echo "$fname Refsnes.
";
}
```

```
familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

פלט:

Jani Refsnes.  
Hege Refsnes.  
Stale Refsnes.  
Kai Jim Refsnes.  
Borge Refsnes.





# שליחת מספר משתנים לפונקציה

---

```
<?php
function familyName($fname, $year) {
 echo "$fname Refsnes. Born in $year

";
}
```

```
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

פלט:

Hege Refsnes. Born in 1975

Stale Refsnes. Born in 1978

Kai Jim Refsnes. Born in 1983



# strict

---

■ PHP מקשר את סוג ה-data עם המשתנה ולכן ניתן לעשות דברים "לא חוקיים" כמו לחבר משתנה מסוג string עם integer כמו בדוגמא

הבאה:

```
<?php
function addNumbers(int $a, int $b) {
 return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5),
and it will return 10
?>
```



# שימוש ב-strict

על מנת למנוע מצב כזה נרצה בדיקה יותר קפדנית של הסקריפט ולכן נגדיר מצב strict ע"י הצהרה על `declare(strict_types=1);` בשורה הראשונה.

```
<?php declare(strict_types=1); // strict requirement

function addNumbers(int $a, int $b) {
 return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer,
an error will be thrown
?>
```



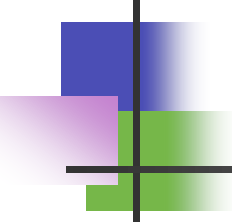
# הגדרת ערכי ברירת מחדל

---

■ ניתן להגדיר בפונקציה משתנים שיקבלו ערך  
default

```
<?php declare(strict_types=1); // strict requirement
function setHeight(int $minheight = 50) {
 echo "The height is : $minheight
";
}
```

```
setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```



# החזרת ערכים

■ על מנת להחזיר ערכים מפונקציה נשתמש ב- return

```
<?php declare(strict_types=1); // strict
requirement
function sum(int $x, int $y) {
 $z = $x + $y;
 return $z;
}
```

```
echo "5 + 10 = " . sum(5, 10) . "
";
echo "7 + 13 = " . sum(7, 13) . "
";
echo "2 + 4 = " . sum(2, 4);
?>
```



# הגדרת ערך מוחזר

- על מנת להגדיר טיפוס נתונים של הערך המוחזר נשתמש ב :  
נקודתיים) בסוף חתימת הפונקציה ונרשום את סוג טיפוס  
הנתונים שנרצה

```
<?php declare(strict_types=1); // strict
requirement
function addNumbers(float $a, float $b) :
float {
 return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```



# Casting

---

- ניתן בצע Casting לערך מוחזר ע"י הצהרה בסוגריים על טיפוס הנתונים אותו נרצה להחזיר

```
<?php declare(strict_types=1); // strict
requirement
function addNumbers(float $a, float $b) : int
{
 return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
?>
```



# עדכון משתנה מקורי

■ כאשר שולחים משתנה לפונקציה, בעצם נשלח העתק של המשתנה ולכן כל שינוי של המשתנה בפונקציה לא משנה את הערך האמיתי של המשתנה מחוץ לפונקציה. על מנת לעדכן את המשתנה המקורי מתוך הפונקציה יש להשתמש ב- &

```
<?php
function add_five(&$value) {
 $value += 5;
}
```

```
$num = 2;
add_five($num);
echo $num;
?>
```





# סוגי משתנים

---

- משתנים יכולי להיות מוגדרים בכל מקום  
בסקריפט
- ישנן 3 סוגי משתנים:
  - לוקאלי- local
  - גלובאלי- global
  - סטאטי- static



# משתנה גלובאלי

---

- משתנה גלובאלי מוגדר מחוץ לפונקציה.
- על מנת לגשת למשתנה מתוך הפונקציה יש להשתמש במילה `global`
- משתנים גלובאליים מאוחסנים בתוך מערך שנקרא `$GLOBALS[index]` ה-`index` מסמל את שם המשתנה



# משתנה גלובאלי מחוץ לפונקציה

■ משתנה שמוגדר מחוץ לפונקציה ויכול להיות בשימוש רק מחוץ לפונקציה

```
<?php
$num = 5; // global scope

function myFunc() {
 // using x inside this function will generate an error
 echo "<p>Variable num inside function is: $num</p>";
}
myFunc();

echo "<p>Variable num outside function is: $num</p>";
?>
```

פלט

- Variable num inside function is:
- Variable num outside function is: 5



# משתנה גלובאלי- שימוש בפונקציה

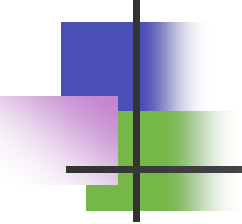
---

- על מנת שפונקציה תוכל להשתמש במשתנה שמוגדר מחוץ לה יש להשתמש במילה `global`

```
<?php
$x = 5;
$y = 10;
```

```
function myTest() {
 global $x, $y;
 $y = $x + $y;
}
```

```
myTest();
echo $y; // outputs 15
?>
```



# משתנה גלובאלי- מערך אינדקסים

■ משתנים גלובאליים נשמרים במערך אינדקסים שנקרא  
\$GLOBALS[index]

```
<?php
```

```
$x = 5;
```

```
$y = 10;
```

```
function myFunc() {
```

```
 $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
```

```
}
```

```
myFunc();
```

```
echo $y; // outputs 15
```

```
?>
```



# משתנה לוקאלי

■ משתנה שמוגדר בתוך הפונקציה, רק הפונקציה מכירה אותו

```
<?php
function myFunc() {
 $num = 5; // local scope
 echo "<p>Variable num inside function is: $num</p>";
}
myFunc();

// using x outside the function will generate an error
echo "<p>Variable num outside function is: $num</p>";
?>
```



# משתנה סטאטי

משתנה סטאטי שומר את הערך האחרון שהוצב לתוכו ולכן ניתן להשתמש בו כמונה. משתנה סטאטי מוגדר על ידי המילה static

```
<?php
```

```
function myFunc() {
 static $x = 0;
 echo $x;
 $x++;
}
```

פלט:

0

1

2

```
myFunc();
myFunc();
myFunc();
```

```
?>
```



---

# **פונקציות לטעינת קבצי קוד (ספריות)**







**חריגות**



# חריגות ב-PHP

- חריגות משמשות על מנת לשלוח מידע על טעות/ התנהגות לא צפויה
- סינטקס:

```
new Exception(message, code, previous)
```

| <u>Parameter</u> | <u>Description</u>                                                                                                                         |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| message          | Optional. A string describing why the exception was thrown                                                                                 |
| code             | Optional. An integer that can be used used to easily distinguish this exception from others of the same type                               |
| previous         | Optional. If this exception was thrown in a catch block of another exception, it is recommended to pass that exception into this parameter |

# דוגמא

```
<?php
function divide($dividend, $divisor) {
 if($divisor == 0) {
 throw new Exception("Division by zero", 1);
 }
 return $dividend / $divisor;
}

try {
 echo divide(5, 0);
} catch(Exception $ex) {
 $code = $ex->getCode();
 $message = $ex->getMessage();
 $file = $ex->getFile();
 $line = $ex->getLine();
 echo "Exception thrown in $file on line $line: [Code $code]
 $message";
}
?>
```

פלט:

Exception thrown in  
/home/TwT0ln/prog.php  
on line 8: [Code 1]  
Division by zero