פרק 15 – פונקציות שימושיות בעבודה עם מערכים

הקדמה

ב PHP יש כ-60 פונקציות שונות לטיפול במערכים. אנחנו כמובן לא ניגע בכולם כי אני רוצה לראות במהלך חיי את פרק מס' 16. אנו ניגע בפונקציות היותר שימושיות, כעשר במספר. מידע על כל הפונקציות לטיפול במערכים ב PHP תוכל למצוא בקישור הבא:

- http://www.php.net/array/

הפרק קצת יותר ארוך מהפרקים הקודמים, אבל הוא קל לבליעה, פשוט תישען על הכסא, תקרא ותתנסה עם PHP תוך כדי קריאה. אל תאמין לאף מילה שאני אומר כאילו היא כתובה באבן – נסה בעצמך, חקור.

הפונקציה is_array

הפונקציה isArray מקבלת כפרמטר משתנה ומחזירה true במידה והמשתנה הוא מערך, אחרת היא מחזירה false. הבט בקטע הקוד הבא:

```
$a = array(1,2,3);
$b = array("a"=>"b", "c"=>"d");
$c = 10;
$d = "Goal!";

if (is_array($a)) echo "a is an array";
if (is_array($b)) echo "b is an array";
if (is_array($c)) echo "c is an array";
if (is_array($d)) echo "d is an array";
```

בתחילת הקוד הגדרנו ארבעה משתנים, מערך בשם $a, מערך משויך בשם $b, משתנה מספרי בשם $c, ומשתנה מסוג מחרוזת בשם $d.

בהמשך ישנם ארבעה משפטי התניה אשר נעזרים בפונקציה is_array בכדי לבדוק האם המשתנה עליהם הם פועלים הוא מערך. כמובן שרק בשני המקרים הראשון הפונקציה is_array תחזיר true והפלט של התוכנית יהיה:

a is an array
b is an array

ישנם מאמצים לבנות פונקציה אשר מחליטה האם מערך מסוים הוא מערך משויך. הפונקציה is_array מחזירה true גם במקרה של מערך נומרי וגם במקרה של מערך משויך. PHP לא מבדילה בין מערך נומרי למערך משויך לכן הפונקציות הללו לא תמיד מחזירות את התשובה שאנו מצפים ממנה להחזיר. הפונקציות אינן חלק מהפונקציות של PHP אלא פותחו על ידי מפתחי PHP כבדים. לא נציג את הגרסאות השונות של הפונקציות הללו כאן.

פונקציות למיון מערכים נומריים

מיון, מיון, מיון זה הדבר ששומעים בשבוע הראשון בכל קורס לאלגוריתמים. בקורס לומדים איך למיין ואיך לעשות את זה בצורה הכי מהירה והכי חסכונית מבחינת זיכרון. ב PHP לא איכפת לנו מזה כי אנחנו לרוב לא מתעסקים עם מיליארד ערכים למיון.

PHP מספקת לנו פונקציה פשוטה ונוחה למיון בשם sort. הפונקציה מקבלת כפרמטר מערך אותו יש למיין. אם המערך מכיל מספרים PHP תמיין את המספרים במערך בסדר עולה, אם המערך מכיל מחרוזות (גם בעברית) המערך ימוין לפי סדר אלפביתי. הבט בדוגמא הבאה:

```
$a = array("אורית" ,"איתי" ,"מורן" ,"גיל");

sort($a);

for($i=0; $i<count($a); $i++)
{
     echo $a[$i]."<br>";
}
```

בשורה השנייה קראנו לפונקציה sort עם המערך $a אותו הגדרנו בשורה הראשונה כמערך של שמות בעברית. הפונקציה sort משנה את המערך $a וממיינת אותו. לאחר מכן נעזרנו בלולאה בכדי להציג את איברי המערך אחד אחרי השני. PHP תציג את איברי המערך בסדר הבא:

אורית, איתי, גיל, מורן

הבעיה עם sort היא שהוא לא ממיין כמו שאתה כנראה מצפה ממנו מחרוזות שיש בהם תווים באותיות קטנות ובאותיות גדולות באנגלית. הבט בקוד הבא:

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il

```
$a = array("Amit", "zoom", "and", "Zoo");

sort($a);

for($i=0; $i<count($a); $i++) {
     echo $a[$i]."<br>";
}
```

בניגוד למה שאתה כנראה מצפה איברי המערך יוצגו על פי הסדר הבא:

Amit, Zoo, and, zoom

PHP כן ממיינת נכונה את האותיות הגדולות בנפרד ואת האותיות הקטנות בנפרד אך היא מתייחסת ל Z כאילו באה לפני a. הסיבה לכך היא האופן בו כל תו משויך למספר, תקן הנקרא ASCII עליו לא נרחיב. כן אומר שגם לבעיה הזו יש פתרון – להמיר את כל המחרוזות במערך לכתב קטן ואז למיין – כמובן שגם לפתרון הזה יש חסרונות. על עבודה עם מחרוזות נלמד בפרקים הבאים.

אם תרצה תוכל למיין את המערך בסדר הפוך – במספרים מהגדול לקטן ובמחרוזות מהאות המתקדמת יותר אחורה אלפביתית. הפונקציה המאפשרת לנו לעשות זאת נקראת rsort (ה r מייצגת את המילה reverse) לדוגמא:

```
$a = array(1, 5, 2, 7, 2, 9, 10, 18);

rsort($a);

for($i=0; $i<count($a); $i++)
{
     echo $a[$i]." ";
}
```

הקוד הבא יציג כפלט: 18, 10, 9, 7, 5, 2, 2, 1.

פונקציות למיון מערכים משויכים

ערכים משויכים ניתן למיין בשני אופנים – על פי המפתחות, או על פי הערכים. מיון על פי מפתח מתבצע בעזרת הפונקציה ksort (ה-k מייצג את המילה key). התחביר של ksort זהה לגמרי לתחביר של sort. כמובן שמיון לפי מפתח שומר על הקשר בין המפתח לערך – הערכים גם הם משנים את מקומם בהתאם לשינוי המיקום של המפתחות הקשורים אליהם.

הפונקציה השנייה – הממיינת לפי ערך נקראת asort. כמובן שגם מיון זה שומר על ההתאמה בין המפתח לערך, אחרת איך נוכל לנצל את התוצאה?!

בקוד הבא מודגם השימוש גם ב ksort וגם ב asort. שים לב כי השתמשנו בלולאת foreach בכדי לעבור על איברי המערך המשויך.

```
$animals = array ("big" => "elephant",
                  "smart" => "dolphin",
                  "cute" => "dog");
asort($animals);

foreach($animals as $key=>$value)
{
     echo "The <b>$value</b> is $key<br>";
}

echo "<p>";
ksort($animals);

foreach($animals as $key=>$value)
{
     echo "The $value is <b>$key</b><br>";
}
```

פלט הקוד הוא:

The **dog** is cute
The **dolphin** is smart
The **elephant** is big

The elephant is **big**
The dog is **cute**
The dolphin is **smart**

שים לב שבפסקה הראשונה הערכים ממוינים לפי הערך – סוג בעל החיים, ובפסקה השנייה הערכים ממוינים לפי המפתח – התכונה של בעל החיים. עוד שים לב כי foreach דאגה להחזיר את מצביע המערך להתחלה לפני כל פעם שרצה על איברי המערך בכדי לוודא שהיא מתחילה מהאיבר הראשון.

כמו מערכים נומריים, גם מערכים משויכים ניתן למיין בסדר הפוך. הפונקציות האחראיות לכך הן krsort הממיינת בסדר הפוך מערך משויך לפי מפתח, ו arsort הממיינת בסדר הפוך לפי ערך.

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il

המרת מערך למחרוזת - implode

PHP מאפשרת לנו לקחת את כל האיברים של מערך נומרי נתון ולהדביק אותם ביחד לכדי מחרוזת. הפונקציה המאפשרת לנו לעשות זאת נקראת implode. הפונקציה מקבלת שני פרמטרים – הראשון הוא התו אשר ישורשר בין כל שני איברים במערך כאשר אלו יועתקו למחרוזת. הפרמטר השני הוא המערך עליו יש לעבוד.

הבט בדוגמא הבאה:

```
$friends = array ("Hanan", "Shmuel", "Aviv", "Yoni");

echo "My friends are: ".implode(" and ", $friends);
```

בשורה הראשונה הגדרנו מערך בעל ארבעה איברים מסוג מחרוזת. בשורה השנייה קראנו ל echo להציג מחרוזת שחלק ממנה הוא המחרוזת המוחזרת מהפונקציה implode. בדוגמא implode מקבלת שני פרמטרים, הראשון הוא מחרוזת " and " (שים לב לרווחים בצידי המילה and). מחרוזת זו תשורשר בין כל שני איברים במחרוזת המתקבלת. הפרמטר השני הוא המערך עליו אנו עובדים והוא במקרה הזה המערך $friends. הפלט של התוכנית נראה כך:

My friends are: Hanan and Shmuel and Aviv and Yoni

הפונקציה implode פועלת גם על מערכים המכילים ערכים מספריים לדוגמא:

```
$numbers = array (18,31,43,27);

echo "The lucky numbers are: ".implode(" - ", $numbers);
```

והפלט יהיה:

The lucky numbers are: 18 - 31 - 43 – 27

כעת בכל פעם שנרצה להציג את איברי המערך לא נצטרך להשתמש בלולאת for אלא פשוט לקרוא ל implode, פשוט נשתמש בקוד הבא:

```
echo implode(" ", $someArray);
```

הקוד הזה יציג לנו את כל איברי המערך $someArray ובין כל שני איברים יוצג תו רווח.

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il

המרת מחרוזת למערך - explode

האחות התאומה של implode היא הפונקציה explode המפרקת מחרוזת למערך. הפונקציה מקבלת שני פרמטרים ומחזירה מערך. הפרמטר הראשון שהיא מקבלת הוא המחרוזת אשר בכל פעם ש explode תראה אותה היא תבין שעד כאן לאיבר הנוכחי, כעת בונים את האיבר הבא. המחרוזת השנייה היא המחרוזת אותה יש לפרק. הבט בקוד הבא:

```
$words = "I think about you every day";

$wordsArray = explode(" ", $words);

echo $wordsArray[2];
```

בשורה הראשונה הגדרנו משתנה מסוג מחרוזת בשם $words ואתחלנו אותו. בשורה השנייה השתמשנו בפונקציה explode – העברנו לפונקציה כפרמטר ראשון את המחרוזת " " (תו רווח), בכך אנו מבקשים מ explode להפריד את המחרוזת לפי תווי הרווח. הפרמטר השני שהוזן ל explode הוא המחרוזת $words שאותה יש לפרק. הפונקציה explode תחזיר מערך בשם $wordsArray.

בשורה השלישית שלפנו את האיבר השלישי של $wordsArray, מכיוון שהפרדנו את המחרוזת למערך על פי הרווחים, בעצם הפרדנו את המחרוזת למילים בודדות (שהרי בין כל שתי מילים יש רווח) ולכן התא השלישי במערך $wordsArray הוא בעצם המילה השלישית, לכן הפלט יהיה המחרוזת "about".

נביט בדוגמא נוספת:

```
$abc = "aababbcbabcabcaababca";

$abcArray = explode("bab", $abc);

echo count($abcArray);
```

בשורה הראשונה בקוד הגדרנו מחרוזת בשם $abc ואתחלנו אותה למחרוזת מטורללת המשלבת את האותיות a,b ו-c. בשורה השנייה נעזרנו ב explode בכדי לפרק את המחרוזת למערך בשם $abcArray. הפרמטר הראשון שהעברנו ל explode הוא המחרוזת על פיה יש לפרק את המחרוזת $abc, המחרוזת "bab" תהווה את המפריד מבחינת explode בין כל שני איברים במערך $abcArray.

במקרה שלנו המערך יראה כך:

aa *bab* **bc** *bab* **cabcaa** *bab* **ca**

כפי שניתן לראות explode מפרקת את המחרוזת על פי המחרוזת bab כך שנוצרים לנו 4 איברים במערך המתקבל – האיבר הראשון הוא מסוג מחרוזת בעל הערך "aa", השני בעל הערך "bc", השלישי בעל הערך "cabcaa" והאיבר האחרון – "ca".

בשורה השלישית אנו מציגים את מספר האיברים במערך בעזרת count ו echo. הפלט יהיה המספר 4, מפני שזה מספר האיברים במערך המתקבל.

מיזוג מערכים – array_merge

הפונקציה array_merge ממזגת שני מערכים או יותר. הפונקציה מקבלת כפרמטר את המערכים אותם יש למזג ומחזירה מערך חדש, ממוזג. הבט בקוד הבא:

```
$a = array("one", "two", "three");
$b = array("four", "five");

$c = array_merge($a, $b);

echo implode(" ", $c);
```

בשתי השורות הראשונות הגדרנו שני מערכים, האחד בשם $a והשני בשם $b ואתחלנו אותם. בשורה השלישית קראנו לפונקציה array_merge עם המערכים $a ו- $b כפרמטרים. הפונקציה החזירה מערך המכיל קודם כל את איברי $a ואחרי כן את איברי $b.

בשורה האחרונה קראנו ל implode, עליה למדנו רק לפני מספר סעיפים, בכדי להציג את איברי המערך החדש $c. הפלט הוא:

one two three four five

במקרה ונרצה למזג מערכים משויכים אנו צריכים להיות מודעים לתכונה של array_merge – אם ישנו אותו מפתח בשני מערכים שונים אז יועתק רק מפתח אחד והערך שיקושר אליו יהיה הערך של האיבר מהמערך האחרון בו מופיע המפתח. הבט בדוגמא הבאה בכדי להבין יותר טוב את התכונה הזו של array_merge:

```
$x = array("a"=>"b", "c"=>"d", "e"=>"f");
$y = array("g"=>"h", "a"=>"t");

$z = array_merge($x, $y);

foreach($z as $key=>$val)
{
     echo "$key - $val<br>";
}
```

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il

ניתן לראות בקוד שהגדרנו שני מערכים $x ו $y כך שלשניהם יש איבר בעל אותו מפתח – "a", במקרה של המערך $x ערך האיבר המשויך למפתח "a" הוא "b" ובמערך $y ערך האיבר המשויך למפתח "a" הוא "t".

בשורה השלישית קראנו ל array_merge עם הפרמטרים $x ו $y, כאשר הסדר חשוב - $x ראשון ו $y שני. את התוצאה הצבנו במערך בשם $z.

לאחר מכן עברנו בעזרת לולאת foreach על כל איברי המערך $z והצגנו אותם שורה אחר שורה. הפלט נראה כך:

```
a - t
c - d
e - f
g – h
```

שים לב שלמרות שיש במערך $x שלושה איברים ובמערך $y שני איברים, במערך הממוזג $z אין חמישה איברים אלא ארבעה, הסיבה לכך היא שישנו איבר משותף מבחינת המפתח לשני המערכים ו array_merge ממזגת (מכאן שמה) את שני האיברים הללו לאיבר אחד. עוד שים לב כי הערך של האיבר התקבל מהמערך האחרון שהכיל אותו, כלומר ה "t" של המערך $y דרס את ה "b" של המערך $x. לכן סדר המערכים חשוב, אם היינו הופכים את הסדר הערך של האיבר המשויך ל "a" היה "b" ולא "t" (כמובן שגם סדר כל האיברים היה משתנה - הנה הפלט של אותה תוכנית רק עם החלפת סדר הפרמטרים לפונקציה array_merge – ראשית $y ורק אחר כך $x:

```
g - h
a - b
c - d
e – f
```

האיבר המשויך ל "a" היה "b" ולא "t" (כמובן שגם סדר כל האיברים היה משתנה - הנה הפלט

הפונקציה array_push

הפונקציה array_push מאפשרת להוסיף מספר איברים בשורת קוד אחת למערך. הפרמטר הראשון ש array_push מקבלת הוא המערך אליו יש להוסיף איברים, שאר הפרמטרים הם ערכי האיברים אותם יש להוסיף לסוף המערך. הפונקציה array_push מחזירה בסיום הרצתה את מספר האיברים במערך החדש. הבט בקוד הבא:

```
$nums = array(4, 1, 6);
array_push($nums, 5, 3, 9, 8);

echo implode(" ", $nums);
```

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il

בשורה הראשונה הגדרנו מערך בשם $nums המכיל שלושה מספרים. לאחר מכן קראנו ל array_push. הפרמטר הראשון לפונקציה הוא המערך $nums – כך אנו אומרים שאנו רוצים לעבוד על המערך הזה. הפרמטרים הנוספים הם האיברים אותם יש להוסיף ל $nums.

בשורה האחרונה הצגנו את ערכי איברי $nums בעזרת echo ו implode.

במידה והינך מעוניין להוסיף רב איבר אחד למערך אל תקרא ל array_push, פשוט הוסף את האיבר "ידנית" מטעמי אופטימיזציה של מהירות.

אם תציין שם מערך כאחד הפרמטרים ל array_push, פרט לפרמטר הראשון, PHP לא תוסיף עבורך את איברי המערך למערך המבוקש אלא תוסיף את המערך כולו כאיבר אחד, כך תקבל מערך דו ממדי. לדוגמא –

```
$nums = array(4, 1, 6);
$moreNums = array(7, 0);
array_push($nums, 5, 3, $moreNums);

echo implode(" ", $nums);
```

בקוד זה הגדרנו שני מערכים - $nums בשורה הראשונה ו $moreNums בשורה השנייה. שניהם מכילים מספרים.

בשורה השלישית נעזרנו ב array_push בכדי להוסיף למערך $nums את האיברים 5, 3 ואת המערך $moreNums. המערך $moreNums יחשב איבר יחיד ב $nums, והמספרים 0 ו- 7 לא ייחשבו איברים במערך $nums, לא באופן ישיר בכל מקרה.

בשורה האחרונה הצגנו את איברי המערך $nums, הפלט נראה כך:

4 1 6 5 3 Array

הפלט הזה דורש הסבר – שלושת האיברים הראשון שהוצגו 4,1 ו-6 הוכנסו ל $nums עוד בשלב האתחול, האיברים 5 ו-3 התווספו ל $nums דרך array_push. האיבר האחרון שהוצג הוא המחרוזת "Array". המחרוזת הזו מוצגת בכל פעם שאנו מנסים להציג ערך של משתנה שהוא מערך, PHP לא תדפיס את איברי המערך אלא תסתפק בלומר – יש לך כאן מערך. אנו נסיק כי array_push הכניסה את המערך $moreNums כאיבר יחיד ל $moreNums.

נשאלת השאלה איך אפשר להציג את איברי תת המערך הזה? ובכן – במקום להיעזר ב echo בצורה עיוורת נוכל לעבור על האיברים של $nums אחד אחד בעזרת לולאת for, ובלפני שנציג את ערך האיבר נבדוק אם מדובר במערך בעזרת הפונקציה is_array שהרחבנו עליה בתחילת הפרק – אם מדובר במשנה רגיל נציג את ערכו, אחרת נציג את איברי תת המערך. הקוד מוצג בעמוד הבא. נעבור על הקוד:

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il

שלושת השורות הראשונות כמעט זהות לשורות בדוגמא הקודמת – אתחול של שני מערכים וקריאה ל array_push בכדי להוסיף איברים למערך בשם $nums. אחד האיברים שהתווסף ל $nums הוא מערך בפני עצמו בשם $moreNums.

שורה לאחר מכן יושבת לולאת for שרצה על כל איברי המערך $nums בעזרת אינדקס בשם $i. בשלב האתחול אתחלנו משתנה מסוג מחרוזת בשם $msg למחרוזת ריקה. הפלט כולו ייאגר במחרוזת זו ונעזר בפקודת echo בסוף הקוד בכדי להציג את ערך המחרוזת. אני רוצה להעיר כאן – כתבתי את הקוד, הרצתי אותו והדפדפן שמר על זכות השתיקה, שיניתי את לולאת ה for הפנימית, שיניתי את החיצונית – לא עזר. בסוף, אחרי בערך 3 דקות של בלבול מוחלט הבנתי ששכחתי להוסיף בסוף הקוד את הפקודה echo, כך יצא שהקוד היה תקין רק שכחתי להציג את הפלט – למד מניסיון של אחרים...

בגוף לולאת ה for אנו רואים משפט התניה הנעזר בפונקציה is_array בכדי לבדוק האם האיבר הנוכחי של המערך $nums הוא מערך, אם כן אז אנו נכנסים לבלוק קוד שמכיל אף הוא לולאה שמיד נעבור עליה, אם לא – כלומר אם האיבר הנוכחי איננו מערך אנו משרשרים את ערכו ל $msg.

במקרה וכן מדובר על מערך אנו רוצים לעבור על כל הערכים שלו בעזרת לולאה אבל לתחום את כל האיברים שנקבל בין התווים { } בכדי להבהיר שמדובר באיבר אחד מסוג מערך שמכיל איברים נוספים (הדבר דומה לתת-קבוצה מתחום מתמטי בשם תורת הקבוצות).

הלולאה הפנימית עוברת על כל איברי המערך הנוכחי, הלא הוא $nums[$i] (האיבר ה $i של $nums הוא מערך אם נכנסו לבלוק של משפט ההתניה). בלולאה אנו משרשרים ל $msg את הערכים של איברי המערך. מכיוון שמדובר במערך דו-ממדי אנו משרשרים שני אינדקסים, האינדקס הראשון הוא $i שאומר לנו שאנו מסתכלים על האיבר ה $i-י של $nums – שהוא בעצמו מערך, האינדקס השני מסתכל בתוך התת-מערך הזה בכדי לקבל את האיבר ה $j שבו.

```
$nums = array(4, 1, 6);
$moreNums = array(7, 0);
array_push($nums, 5, 3, $moreNums, 9);

for($i=0, $msg=""; $i<count($nums); $i++)
{
     if (is_array($nums[$i]))
     {
          $msg .= " { ";
          for($j=0; $j<count($nums[$i]); $j++)
          {
               $msg .= $nums[$i][$j]." ";
          }
          $msg .= " } ";
     }
     else $msg .= $nums[$i]." ";
}

echo $msg;
```

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il

הפלט נראה כך:

4 1 6 5 3 { 7 0 } 9

אם לתת המערך היה תת מערך משל עצמו לא היינו רואים את ערכיו כי אנחנו נכנסים רק רמה אחת לעומק, נוכל להוסיף עוד קוד שיגיע לרמת עומק נוספת אבל הדרך הטובה ביותר להתמודד עם כל רמת עומק, לפחות באופן תיאורטי, היא בעזרת טכניקה הנקראת רקורסיה עליה לא נלמד בפרק זה.

הפונקציה array_pop

כמו שסבתא שלי אומרת - לכל סיר יש מכסה (והיא יודעת – יש לה מטבח 100 מ"ר), וכמו שאני נוהג להגיד (למען האמת זו הפעם הראשונה) לכל push יש pop, ו PHP אינה יוצאת דופן. הפונקציה array_pop מושכת את האיבר האחרון ממערך המועבר אליה כפרמטר, מחזירה את ערכו, ומסירה אותו מן המערך. כהרגלנו, הבט בקוד:

```
$todoList = array("eat", "shower", "finish chapter 15");

$iDid = array_pop($todoList);

echo "I did $iDid <br>";
echo "I still need to: ". implode(" and ",$todoList);
```

בשורה הראשונה הגדרנו מערך בשם $todoList שאיבריו מהווים מחרוזות המציינות את כל מה שאני צריך לעשות היום.

בשורה השנייה קראנו ל array_pop עם הפרמטר $todoList. הפונקציה תחזיר את האיבר האחרון לתוך משתנה בשם $iDid ותסיר אותו מן המערך $todoList.

השורות שלאחר מכן מציגות את תוכן המשתנה $iDid ואת איברי המערך $todoList (שוב בעזרת implode). הפלט נראה כך:

I did finish chapter 15
I still need to: eat and shower

במקרה ואין עוד איברים במערך, הפונקציה array_pop תחזיר את הערך NULL שמבחינתו זהה לערך false כך שנוכל להשתמש ב array_pop בתוך תנאי לולאה. בעצם זו דרך נוחה להציג את כל איברי המערך בסדר הפוך, כמובן שהמערך יתרוקן בסוף הלולאה. נסתכל על דוגמא לקוד שכזה בשילוב עם הפונקציה array_shift בסעיף הבא.

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il

הפונקציה array_shift

תפקיד הפונקציה array_shift דומה לתפקיד array_pop, רק שהאיבר שהיא מסירה ומחזירה איננו האיבר האחרון אלא האיבר הראשון במערך. גם array_shift תחזיר false כאשר לא יוותרו ערכים במערך.

הבט בקוד הבא המשלב באופן מעניין את array_shift ואת array_pop:

```
$a = array (1, 2, 3, 4, 5, 6, 7, 8);

while(true)
{
     $n = array_shift($a);

     if ($n == false) break;
     echo $n." ";

     $n = array_pop($a);

     if ($n == false) break;
     echo $n. " ";
}
```

הרעיון מאחורי הקוד הזה הוא להציג את איברי המערך באופן הבא – נציג את האיבר הראשון, ונסיר אותו, לאחר מכן נציג את האחרון ונסיר אותו. נמשיך כך עד שלא יישארו איברים במערך. בשורה הראשונה הגדרנו מערך בשם $a עם הערכים המספריים 1 עד 8.

בשורה השנייה אנו נכנסים ללולאת while עם שהתנאי שלה הוא true – כלומר אנו נכנסים ללולאה שלא תסתיים אף פעם שהרי תנאי הלולאה תמיד יוערך כאמת, הרי אי אפשר לקוות ליותר אמת מ true. בכל זאת הלולאה הזו תסתיים בעזרת השימוש ב break כפי שנראה בהמשך.

בשורה הראשונה בגוף הלולאה אנו קוראים ל array_shift עם המערך $a ובכך מושכים את ערך האיבר הראשון במערך $a למשתנה עזר שנקרא $n. שורה אחרי כן אנו בודקים אם $n הוא false, כלומר אם array_shift סימנה לנו שאין איברים במערך, במקרה כזה אנו יוצאים מהלולאה בעזרת מילת המפתח break שהרי סיימנו את תפקידנו – המערך ריק.

שורה אחרי כן אנו קוראים לפונקציה array_pop על המערך $a ומבצעים בדיוק את אותו הקוד – מחזירים את ערך האיבר ל $n, בודקים אם הוא false – במידה וכן יוצאים מהלולאה בעזרת break. אם $n אינו false אנו מציגים את ערכו.

הפלט של הקוד נראה כך:

```
1 8 2 7 3 6 4 5
```

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il

יש רק בעיה אחת עם הקוד הזה שלא עולה עם המערך הנוכחי. נניח ונחליף את האיבר שערכו 3 בערך 0 במערך ונריץ את הקוד. הפלט, למרבה ההפתעה, יראה כך:

1 8 2 7

מוזר, יכולתי להישבע שאמורים להיות 8 איברים במערך.. איפה כולם הלכו? ובכן, כאשר קראנו לפונקציה array_pop או array_shift והיא החזירה לנו את הערך 0, ואת הערך הזה אנו השווינו ל false אנו מקבלים פסוק אמת ומילת המפתח break מתבצעת. זו בעיה כי false מוערך כ-0. הדרך לפתור אותה היא בעזרת האופרטור ===. אופרטור זה משווה בין שני משתנים אך לא רק בין הערכים שלהם אלא גם בין הטיפוסים שלהם. במקרה הזה 0 שונה מ false כי הראשון הוא טיפוס מספרי והשני בוליאני. אם נחליף את סימני ההשוואה == ב === בקוד (יש צורך ב-2 החלפות כאלו – אחת ל array_pop ואחת ל array_shift) הקוד יתבצע כראוי. דבר נוסף שעלינו לשנות בקוד הוא את false ל NULL. הפונקציות array_pop ו array_shift מחזירות NULL שהוא שונו מבחינת הטיפוס מ false, המילה השמורה NULL איננה טיפוס בוליאני. הנה הקוד המתוקן:

```
$a = array (1, 2, 3, 4, 5, 6, 7, 8);

while(true)
{
     $n = array_shift($a);

     if ($n === NULL) break;
     echo $n." ";

     $n = array_pop($a);

     if ($n === NULL) break;
     echo $n. " ";
}
```

והפלט נראה כך:

1 8 2 7 0 6 4 5

סיכום

בפרק זה למדנו על פונקציות רבות, אך אלו רק הבסיסיות ביותר לעבודה עם מערכים. אני יכול להרגיע אותך – אני לא זוכר את התפקיד של כולן ואני חוזר המון פעמים ל php.net בכדי לבדוק איך בדיוק פונקציה פועלת והאם קיימת פונקציה המבצעת את מה שאני צריך לבצע. אם לא – כותבים אותה, אבל על כתיבת פונקציות נדבר עוד מספר פרקים מעכשיו. בפרק נתקלנו גם באופרטור === שמתברר כמאוד שימושי במצבים מסוימים. בפרק הבא נלמד כיצד ליצור פונקציות משלנו בכדי לייעל את העבודה ב PHP.

כל הזכויות שמורות לגיל כהן, http://www.flashoo.co.il