פרק 14 – מערכים משויכים

הקדמה

מערך משויך הוא מערך ככל מערך, אלא שהאינדקס שלו איננו מספרי אלא מחרוזת. כלומר – בכדי לגשת לאיבר במערך לא צריך לדעת את מספר אלא מחרוזת. המחרוזת משמשת כמפתח לקבלת המידע או הערך של האיבר. בגלל זה קוראים למערך משויך גם בשם מפתח-ערך. אבל רגע, כבר נתקלנו בזה בערך אלף פעם – ככה שולחים מידע לפלאש – שם-ערך.

המונח שם ומפתח הם זהים ותראה שאני משתמש בהם לסירוגין לאורך הפרק.

הגדרת מערך משויך, גישה לערכיו – ולמה זה טוב?

נביט בהגדרה של מערך משויך ב PHP:

```
$profile = array ("name" => "Gil Cohen",
                  "age" => 21,
                  "sex" => "male",
                  "occupation" => "student");

echo $profile["name"];
```

בארבעת השורות הראשונות הגדרנו מערך משויך בשם $profile. כמו כל מערך הגדרתו מתחילה בשם (לא לשכוח סימן $), סימן הצבה אם רוצים לאתחל את המערך, מילת המפתח array האומרת ל PHP כאן נבנה מערך ואז בניגוד למערכים רגילים אנו מזינים מחרוזת את המפתח (או השם) של האיבר הראשון, מלווה בסימן חץ (=>) שמורכב מסימן השוויון ומהסימן גדול מ ואחריו מופיע הערך.

האיבר הראשון שהגדרנו הוא מסוג מחרוזת. שם האיבר או המפתח הוא "name" וזו המחרוזת שנצטרך להציב בין הסוגריים המרובעים בכדי למשות את הערך "Gil Cohen". מיד אחרי הגדרת האיבר הראשון השתמשנו בפסיק בכדי להפריד בינו לבין האיבר השני. מעבר השורה הוא אופציונאלי ואיננו חובה, אך זו דרך טובה לכתוב קוד קריא.

האיבר השני הוגדר אף הוא בפורמט מפתח-ערך, המפתח הוא "age" והערך הוא 21. שים לב כי הערך הוא ערך מספרי – תוכל לשלב במערכים ב PHP טיפוסים שונים בתוך מערכים.

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

האיבר השלישי שהוגדר שמו הוא "sex" וערכו הוא "male", והאיבר הרביעי והאחרון – שמו הוא "occupation" וערכו "student". לבסוף תחמנו את הגדרת המערך בסוגר ובליווי נקודה פסיק.

בשורה שאחרי הגדרת המערך נעזרנו ב echo בכדי להציג את ערכו של האיבר ששמו "name" מהמערך $profile. שים לב שבמקום להשתמש במספר השתמשנו במחרוזת. יש לכך יתרון גדול. הבט בקוד הבא המבצע בדיוק את אותו הדבר אך בעזרת מערכים רגילים (או בשמם הפחות מוכר – מערכים נומריים [נומרי = מספרי]):

```
$profile = array("Gil Cohen", 21, "male", "student");

echo $profile[0];
```

אין ספק – הקוד הזה קצר יותר אבל שים לב לשורת הקוד השנייה – אנו מבקשים להציג את שם המשתמש אך 0 לא רומז כלל שמודבר כאן על שם של מישהו. כאשר תחזור לקוד הזה לאיתור באגים, עדכון תוכנה או סתם לצורכי נוסטלגיה לא תבין כלל מה ה-0 הזה מביע. יותר מכך – בדרך כלל אנו לא באמת מגדירים מערכים בקוד אלא מושכים שורה ממסד נתונים. מה יקרה אם יום בהיר אחד האחראי על מסד הנתונים יחליט שהשם יופיע אחרי הגיל? אז נצטרך לעדכן את כל הקוד. בעזרת מערכים משויכים אנו מקבלים הפרדה חזקה יותר בין שכבת המידע לשכבת הקוד ועל הדרך משאירים את הקוד קריא יותר.

הוספת איבר ושינוי ערך איבר של מערך משויך

הוספה של איבר למערך משויך דומה להוספת איבר במערך רגיל, הבט בקוד הבא:

```
$meals = array("breakfast" => "eggs",
               "launch" => "hamburger",
               "dinner" => "toast");

$meals["snack"] = "ice-cream";
```

בשורת 1-3 הגדרנו מערך משויך חדש בשם $meals שמכיל שלושה איברים המהווים את שלושת הארוחות ביום. בשורה הרביעית הוספנו למערך $meals איבר חדש שהמפתח שלו הוא "snack" וערכו הוא "ice-cream".

מהקוד רואים כי כל שיש לעשות בכדי להוסיף איבר חדש למערך הוא להחליט על מפתח ועל ערך ולהשתמש בתחביר המוכר מהוספת איבר למערך נומרי, רק שהמפתח איננו מספרי אלא מסוג מחרוזת.

שינוי ערכו של איבר מתבצע בדיוק באותו אופן: ראשית כותבים את שם המערך, לאחר מכן את הסוגריים המרובעים אשר ביניהם יושב המפתח של האיבר אשר את ערכו אנו רוצים לשנות. השורה הבאה לדוגמא תחליף את ארוחת הצהריים שלנו בפסטה

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

```
$meals[“launch”] = “pasta”;
```

מערכים משויכים דו-ממדיים

נניח שברצוננו לאחסן מספר פרופילים, נוכל לעשות זאת בעזרת מערך משויך המקונן בתוך המערך נומרי. הבא בקוד הבא:

```
$profiles = array (

              array ("name" => "Gil Cohen",
                     "age" => 21,
                     "sex" => "male" ),

              array ("name" => "Orit Cohen",
                     "age" => 22,
                     "sex" => "female"),

              array ("name" => "Dotan Porat",
                     "age" => 23,
                     "sex" => "male") );
```

בקוד ניתן לראות הגדרה של מערך נומרי שכל אחד מאיבריו מהווה מערך משויך בפני עצמו. כל מערך משויך מהווה פרופיל של אדם אחר.

בכדי לגשת לדוגמא לשם של המשתמש השני נוכל להיעזר בקוד:

```
echo $profiles[1]["name"];
```

תא מספר 1 הוא בעצם התא השני – כך אנו ניגשים לנתונים של האדם השני. אנחנו כעת בתוך מערך משויך, כעת אנו צריכים להציב בסוגריים נוספים (כי מדובר בערך דו-ממדי ועל כן צריך שני אינדקסים) את המפתח. הקוד הזה יציג את המחרוזת "Orit Cohen" לדפדפן.

טיול על מערך משויך

יש לנו בעיה – אנחנו לא יכולים לרוץ, בעזרת הכלים הנוכחיים שלנו, בלולאה על כל הערכים כפי שיכולנו לעשות עם מערכים נומריים. אנו צריכים כלי חדש. ובכן אנו נלמד על שלושה פונקציות מאוד שימושיות בעבודה עם מערכים משויכים.

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

שתי הפונקציות הראשונות שנפגוש פועלות יחד בכדי לטייל על מערך משויך – הפונקציות נקראות each ו- list. אנו ראשית נלמד כיצד each פועלת, כיצד list פועלת ואז נשלב אותן לצרכינו.

הפונקציה השלישית – היא בעצם לולאה בשם foreach שמהווה אלטרנטיבה לשימוש ב each ו list. השימוש ב foreach נפוץ ונוח יותר אך בכדי להבין את דרך פעולתה אני מאמין שצריך להבין איך each ו list פועלות.

הפונקציה each

לכל מערך ב PHP יש מה שנקרא מצביע מערך. מצביע מערך מצביע על התא הנוכחי במערך. התא הנוכחי תלוי באילו פקודות השתמשנו, אבל כעיקרון הפעולה שלו שקופה מבחינתנו כמפתחים. הפונקציה each מקבלת כפרמטר מערך משויך ומסתכלת על מצביע המערך בכדי לראות על איזה תא צריך לעבוד כרגע.

מה ש each עושה היא להחזיר מערך משויך חדש עם המפתחות "key" ו- "value". הערך של האיבר "key" יהיה המפתח של התא הנוכחי במערך שהוזן ל each כפרמטר, הערך של "value" יהיה הערך של אותו תא. נביט בדוגמא:

```
$fav = array ("color" => "blue", "season" => "fall");

$favElement = each($fav);

echo "My favorite ". $favElement["key"];
echo " is ". $favElement["value"];
```

בשורה הראשונה הגדרנו מערך משויך בשם $fav. המערך מכיל שני איברים – צבע אהוב ועונה אהובה. שורה אחרי זה אנו רואים שימוש בפונקציה each. הפונקציה מקבלת כפרמטר את המערך המשויך $fav ומחזירה מערך משויך חדש, למערך המשויך החדש קראנו $favElement. מערך זה מכיל שני איברים – המפתחות של האיברים הללו תמיד יהיו "key" ו- "value", הערכים של המפתחות הללו נלקחים מהאיבר הנוכחי במערך $fav. מכיוון שזו הפעם הראשונה שקראנו ל each אז הערך של $favElement באיבר "key" יהיה "color" והערך באיבר "value" יהיה "blue".

בשתי השורות שלאחר הקריאה ל each הצגנו את ערכי איברים אלו. הפלט לדפדפן יהיה: My favorite color is blue.

אם נקרא ל each פעם נוספת כאשר $fav יהיה הארגומנט שלה, אנו לא נעבוד יותר על האיבר הראשון של $fav אלא על האיבר השני. הפונקציה each דואגת לקדם את מצביע המערך למקום המתאים.

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

אם each לא רואה עוד איברים – כלומר אם each הגיעה לסוף המערך היא לא מחזירה מערך משויך אלא את הביטוי הבוליאני false. הדבר לא נעשה במקרה – בזכות התכונה הזו של each נוכל לכתוב את כל איברי המערך בבת אחת בעזרת לולאת while. הבט בקוד הבא:

```
$fav = array ("color" => "blue",
              "movie" => "Good Will Hunting",
              "season" => "fall" );

while($favElement = each($fav))
{
     echo "My favorite ". $favElement["key"];
     echo " is ". $favElement["value"];
     echo "<br>";
}
```

גם בקטע הקוד הזה אנו נתקלים במערך $fav רק שעכשיו הוא מכיל שלושה איברים. מיד אחרי הגדרת המערך יושבת לולאת while. בתוך תנאי הלולאה אנו רואים קריאה לפונקציה each על המערך $fav. הפונקציה כאמור תחזיר את הערך false כשנגיע לסוף המערך כך אנו יכולים להיות בטוחים שהלולאה תסתיים בדיוק בזמן. כל עוד לא הגענו לסוף המערך each תחזיר מערך משויך כפי שתואר קודם, למערך זה קראנו $favElement.

בגוף לולאת ה while הצגנו בעזרת echo את האיבר הנוכחי של $fav, איבר זה ערכיו נמצאים ב $favElements כערכים של מערך משויך על פי המפתחות "key" ו- "value".

אבל רגע – אם מצביע המערך הגיע לסופו אז לא נוכל להשתמש יותר ב each כי הוא תמיד יחזיר false. זו באמת בעיה – each דומה ללקוח מרגיז בספריית וידאו – הוא אף פעם לא דואג להחזיר את הסרט אחורה להתחלה. לכן, לפני שאנו רוצים להציג את כל הערכים במערך זה יהיה רעיון חכם להחזיר את מצביע המערך לתחילת המערך בעזרת פונקציה בשם reset. הפונקציה reset מאפסת את מצביע המערך ובכך מבטיחה לנו שאנו נציג את המערך מתחילתו. הקוד הבא הוא קוד נכון יותר, מהקוד הנאיבי שהוצג קודם לכן שהניח שמצביע המערך מאופס.

```
$fav = array ("color" => "blue",
              "movie" => "Good Will Hunting",
              "season" => "fall" );

reset($fav);

while($favElement = each($fav))
{
     echo "My favorite ". $favElement["key"];
     echo " is ". $favElement["value"];
     echo "<br>";
}
```

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

שים לב לשימוש בפונקציה reset, מיד לפני הלולאה. הפונקציה מקבלת כפרמטר את המערך אשר את המצביע שלו יש לאפס.

מערכים משויכים דורשים קצת הסתגלות. אני מציע שתנסה לשחק עם מערך משויך ועם הפקודה each בכדי להיות בטוח שאתה שולט בנושא לפני שתמשיך לסעיף הבא.

הפונקציה list

הפונקציה list מפרקת מערך נומרי (כלומר מערך לא משויך) למשתנים מבודדים. לפני שנענה על השאלה למה אנחנו רוצים לעשות דבר כזה, נראה איך עושים את זה. הבט בקוד הבא:

```
$names = array("Eti", "Roni", "Yael");

list($name1, $name2, $name3) = $names;

echo $name1."<br>";
echo $name2."<br>";
echo $name3."<br>";
```

בשורה הראשונה הגדרנו מערך בשם $names בעל שלושה איברים. בשורה השנייה קראנו לפונקציה list (זו לא באמת פונקציה אלא פקודה מובנת ב PHP) עם שלושה ארגומנטים, אחריה הצבנו את סימן ההשמה (=) ואחריו את המערך אותו יש לפרק. (אכן תחביר מוזר).

הפקודה list תיקח את הערך הראשון במערך ותכניס אותו ל $name1, את הערך השני תכניס ל $name2 ואת השלישי ל $name3. שלושת השורות האחרונות בקוד מוודאות האם באמת הערכים של $name1, $name2 ו- $name3 הם המחרוזות "Roni", "Eti" ו- "Yael" בהתאמה.

הפקודה list שימושית כאשר מקבלים מערך נומרי ממקור חיצוני כמו מסד נתונים, ורוצים לתת משמעות לערכים. לדוגמא, נניח שאנו מקבלים שורה של טבלה ממסד נתונים לתוך מערך בשם $userInfo. התא הראשון של המערך הוא מספר המשתמש, התא השני הוא שם המשתמש והתא השלישי מכיל את כתובת המייל של המשתמש. הקוד הבא מאפשר לנו לקחת את המערך ולעבוד בצורה נוחה יותר עם הנתונים:

```
list (, $userName, $email) = $userInfo;
```

אחרי שורת קוד זו נוכל לגשת לשם המשתמש בעזרת המשתנה $userName ולמייל של המשתמש בעזרת המשתנה $email.

שים לב שלא הגדרתי משתנה עבור האיבר הראשון שהוא מספר המשתמש – זאת מפני שאני לא חייב. אם לא מעוניינים באחד האיברים פשוט משאירים מקום ריק במקום בו היה אמור להיות

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

המשתנה שיכיל את הערך של האיבר. אם לדוגמא כל שהייתי רוצה היה את המייל ואת מספר המשתמש הייתי כותב:

```
list ($userID , , $email) = $userInfo;
```

פשוט דילגנו מעל האיבר האמצעי המייצג את שם המשתמש.

שילוב each ו- list

השילוב של each ו-list הוא לא המקרה היחיד בו תשתמש בפקודות הללו אך הן משתלבות יפה בכדי לטייל על מערכים משויכים.

ניזכר בקוד שראינו בסעיף על הפונקציה each המאפשר לנו לטייל על מערך משויך:

```
$fav = array ("color" => "blue",
              "movie" => "Good Will Hunting",
              "season" => "fall" );
reset($fav);
while($favElement = each($fav))
{
     echo "My favorite ". $favElement["key"];
     echo " is ". $favElement["value"];
     echo "<br>";
}
```

מה רע בקוד הזה? מה list יכולה להוסיף שחסר כאן, הרי הקוד מבצע את מה שאנו מצפים ממנו לבצע. ובכן, list מאפשרת לתת משמעות לשמות המשתנים. המערך $favElements עם המפתחות "key" ו "value" לא רומז לנו על משמעות הנתונים.

אמרנו כבר ש each מחזירה מערך משויך אבל זה לא מדויק – each מחזירה מערך בעל 4 איברים – שניים מהם עם המפתחות "key" ו "value" כפי שלמדנו ושניים מהם עם המפתחות 0 ו-1 מה שהופך את המערך ש each מחזירה למערך חצי-נומרי.

עוד אמרנו שהפקודה list לוקחת מערך נומרי ומפרקת אותו, לכן נוכל לפרק את המערך ש each מקבלת ולפרק אותו למשתנים עם שמות בעלי משמעות. הבט בדוגמא הבאה:

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

```
$family = array ("mother" => "Yael",
                 "father" => "Ziv",
                 "brother" => "Shavit" );

reset($family);

while(list($relation, $name) = each($family))
{
     echo "My $relation is $name<br>";
}
```

בשורה הראשונה הגדרנו מערך המייצג משפחה. המערך הוא מערך משויך עם המפתחות "father", "mother" ו- "brother" והערכים "Ziv", "Yael" ו- "Shavit" בהתאמה.

בשורה השנייה איפסנו את מצביע המערך, זהו הרגל טוב – לא כדאי להניח שהמצביע בהתחלה, ואם רוצים לעבור על כל איברי המערך אנו צריכים להחזיר "ידנית" בעזרת reset את המצביע.

שאר הקוד מורכב מלולאת while. בתנאי הלולאה אנו קוראים לפונקציה each עם המערך $family. הפונקציה מחזירה מערך המכיל את המפתח ואת הערך של האיבר הנוכחי עליו אנחנו עובדים ב $family והמערך הזה מועבר ל list אשר מפרקת אותו למשתנים $relation ו $name. המשתנה הראשון, $relation, יקבל בריצה הראשונה של הלולאה את המחרוזת "mother" והמשתנה $name יקבל את המחרוזת "Yael". בריצה השנייה של הלולאה each תחזיר ל list מערך משויך שמהווה את האיבר השני ב $family, הפקודה list תפרק אותו שוב ל $relation ו- $name, הפעם ערכו של $relation יהיה המפתח של האיבר השני במערך $familty, כלומר ערכו יהיה "father" וערכו של $name יהיה "Ziv".

בגוף לולאת ה while ישנה פקודת echo שמדפיסה את הקרבה של אדם ואת שמו. הפלט של התוכנית יהיה:

My mother is Yael
My father is Ziv
My Brother is Shavit

יותר נוח לעבוד עם משתנים בשם $name ו $relation מאשר עם $favElement["key"] ו $faveElement["value"]. זה בעצם הערך המוסף של list בשילוב עם each.

יש חדשות טובות – ישנה פקודה בשם foreach המאפשרת לטייל על מערך משויך באופן פשוט יותר מאשר עם הפונקציה echo והפקודה list. בסעיף הבא נראה איך משתמשים בה. אם אתה שואל את עצמך – "אז למה הוא שיגע לי את השכל עם הפקודות הללו אם בכלל לא משתמשים בהן?" ובכן – אמנם בהקשר של מערכים משויכים each ו list פחות שימושיות אבל אלו פקודות מאוד שימושיות בעבודה עם מסדי נתונים. בנוסף, אי אפשר ממש להבין איך foreach פועלת מבלי לראות איך each ו list פועלות.

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

לולאת foreach

הפקודה foreach היא מן שילוב קסום של reset, each ו list ביחד. הפקודה foreach היא בעצם לולאה שעוברת על כל הערכים של מערך משויך, ואפילו דואגת לפני המעבר לקרוא ל reset באופן אוטומטי.

הבט בקוד הבא, המבצע בדיוק את מה שביצענו קודם לכן בעזרת reset, list ו each, הפעם בגסה ה foreach-ית שלו:

```
$family = array ("mother" => "Yael",
                 "father" => "Ziv",
                 "Brother" => "Shavit" );

foreach($family as $relation => $name)
{
     echo "My $relation is $name<br>";
}
```

ראשית הגדרנו את המערך $family בדיוק כפי שהוגדר בדוגמא הקודמת. אחרי הגדרת המערך אנו רואים את הפקודה foreach מלווה בסוגריים עגולים. בתוך הסוגריים העגולים אנו כותבים את המערך עליו אנו רוצים לטייל בדוגמא שלנו אנו רוצים לטייל על $familty. לאחר שם המערך אנו כותבים את המילה השמורה as, זהו חלק מהתחביר. לסיום אנו כותבים את המשתנה שיכיל את המפתח של האיבר הנוכחי מלווה בסימן החץ ואחריו את שם המשתנה שיכיל את הערך של האיבר הנוכחי.

שוב – foreach, פותחים סוגריים, כותבים את שם המערך, המילה השמורה as, לאחר מכן את שם המשתנה שיכיל את המפתח של האיבר הנוכחי, החץ המוכר מהגדרת המערכים המשויכים, ואז שם המשתנה שיכיל את ערכו של האיבר הנוכחי.

התחביר של foreach דורש הסתגלות, אבל אחרי שמתרגלים אליו הוא קל מאוד לשימוש ושימושי מאוד, בעיקר בעבודה עם מסדי נתונים.

בגוף הלולאה השתמשנו ב $relation ו $name כאילו היו משתנים רגילים לגמרי, מבלי לדאוג למעבר לאיבר הבא – foreach פשוט רצה על כל האיברים במערך $family. הבט בדוגמא נוספת בכדי להיות בטוח שהבנת:

```
$menu = array ("Pizza" => 41,
               "Sandwitch" => 21,
               "Toast" => 38,
               "Pasta" => 57 );

foreach ($menu as $order => $cost) {
     echo "We have $order in $cost NIS<br>";
}
```

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

הקוד בדוגמא הזו כמעט זהה לקוד בדוגמא הקודמת. גם כאן יש לנו מערך בשם $menu המייצג תפריט במסעדה. המערך הוא מערך משויך כך שהמפתח הוא שם ההזמנה והערך הוא מחיר ההזמנה.

אחרי הגדרת המערך אנו רואים לולאת foreach אשר רצה על המערך $menu. הלולאה מעבירה את המפתח של האיבר הנוכחי במערך למשתנה $order (שם ההזמנה) ואת הערך של האיבר הנוכחי למשתנה $cost (מחיר ההזמנה). בגוף הלולאה אנו מציגים בעזרת echo את התפריט לדפדפן הרעב.

קינון foreach-ים

כבר ראינו שניתן לקונן לולאה בתוך לולאה. foreach היא לולאה ככל לולאה וגם אותה ניתן לקונן בתוך כל לולאה אחרת. נחזור לקוד שראינו בעבר – פרופילי משתמשים:

```
$profiles = array (

              array ("name" => "Gil Cohen",
                     "age" => 21,
                     "sex" => "male" ),

              array ("name" => "Orit Cohen",
                     "age" => 22,
                     "sex" => "female"),

              array ("name" => "Dotan Porat",
                     "age" => 23,
                     "sex" => "male") );
```

כעת נעבור על כל המערך בעזרת לולאת for ועל כל אחד מהאיברים שלו – שהם בעצם בעצמם מערכים משויכים, נעבור בעזרת foreach. הקוד להצגת המערך הקודם יראה כך:

```
for($i=0; $i<count($profiles); $i++)
{
     echo "<b>User $i:</b><br>";
     foreach($profiles[$i] as $property => $value)
     {
          echo "$value, ";
     }
     echo "<p>";
}
```

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

התחלנו בלולאת for מוכרת שרצה על המערכת הנומרי $profiles. כל איבר במערך $profiles הוא מערך משויך בפני עצמו ולכן בתוך לולאת ה for קיננו לולאת foreach שתטפל באיבר הנוכחי של $profiles, כלומר במערך המשויך הנוכחי.

המערך בו אנו רוצים לטפל בכל שלב ב foreach הוא, כאמור, האיבר הנוכחי ב $profiles, כלומר המערך הנוכחי הוא $profiles[$i] כאשר $i הוא האינדקס שרץ מ-0 עד ל-2 (או ליתר דיוק עד ל count($profiles) שרק בדוגמא הזה, במקרה, ערכו 2).

הפלט לדפדפן נראה כך:

User 0:
Gil Cohen, 21, male,

User 1:
Orit Cohen, 22, female,

User 2:
Dotan Porat, 23, male,

סיכום

פרק זה לא היה פרק קל, למדנו על הפונקציה echo, על הפקודה list, על הפקודה reset וקינחנו בלולאה חדשה בשם foreach. מערכים משויכים זה לא נושא מסובך אבל הוא דורש הסתגלות והסתגלות היא לא רק דרך קריאה אלא בעיקר דרך כתיבה. כתוב, התנסה, כנס ל http://www.php.net ותראה מה יש להם להגיד על הפקודות הללו. אני לא חשפתי את כל הפרטים מאחורי הפקודות הללו מפני שאחרת היינו סוטים מהנושא, אבל אחרי שתתרגל ותשלוט בנושא המערכים המשויכים, אזור אומץ, כנס ולמד עוד. בפרק הבא נירגע ונסתכל על פונקציות קלות לתפעול ומאוד שימושיות של PHP יש להציע בעבודה עם מערכים.

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