מערכים

יצירת מערך

ב-PHP יוצרים מערך חדש באמצעות הפקודה `array` אשר מחזירה `reference` למערך החדש. ניתן לשלוח לפקודה הזו את הערכים שבהם רוצים לאתחל את המערך החדש שיוצרים. לכל ערך במערך החדש יש `key` שידוע גם כמספר האינדקס.

הדוגמא הבאה מציגה את יצירתו של מערך פשוט אשר מאותחל בערכים 100, 200 ו-300.

```
<?php
$vec = array(100,200,300);
for($index=0; $index<3; $index++)
{
      echo "<BR>vec[$index]=".$vec[$index];
}
?>
```

הפלט שנקבל הוא:

```
vec[0]=100
vec[1]=200
vec[2]=300
```

כיוון שה-`type` המדוייק של כל אחד מערכי המערך נקבע בעת היצירה שלו ניתן באותו אופן לייצור גם מערך של מחרוזות תווים.

```
<?php
$vec = array("moshe","david","mike");
for($index=0; $index<3; $index++)
{
      echo "<BR>vec[$index]=".$vec[$index];
}
?>
```

הפלט שנקבל הוא:

```
vec[0]=moshe
vec[1]=david
vec[2]=mike
```

כאשר יוצרים מערך חדש ניתן לקבוע כ-`keys` ערכים אחרים במקום מספרי האינדקס המקובלים. עושים זאת באמצעות האופרטור => תוך מיקומו בין ה-`key` (משמאל) וה-`value` (מימין). ה-`key` לא חייב להיות ערך מספרי שלם.

הדוגמא הבאה מציגה יצירת מערך שמחזיק (כ- values) שמות של מדינות ובתור ה-keys שלהם הוא מחזיק בשמות הקיצור שלהם.

```
<?php
$vec = array("IL"=>"Israel","USA"=>"United States","UK"=>"United Kingdom");
echo "<BR>".$vec["IL"];
echo "<BR>".$vec["USA"];
echo "<BR>".$vec["UK"];
?>
```

הפלט שנקבל הוא:

```
Israel
United States
United Kingdom
```

הפונקציה var_dump

באמצעות הפונקציה var_dump ניתן להדפיס את תוכנו של מערך (או של כל ערך composite מסוג אחר... כגון אובייקט שנוצר ממחלקה). הדוגמא הבאה מציגה קריאה להפעלת הפונקציה לצורך הדפסת תוכנו של מערך.

```
<?php
$vec = array("IL"=>"Israel","USA"=>"United States","UK"=>"United Kingdom");
var_dump($vec);
?>
```

הפלט שנקבל הוא:

```
array(3) { ["IL"]=> string(6) "Israel" ["USA"]=> string(13) "United States"
["UK"]=> string(14) "United Kingdom" }
```

הפונקציה var_dump מאפשרת להדפיס את תוכנם של מספר מערכים. הדוגמא הבאה מציגה זאת.

```
<?php
$countries = array("IL"=>"Israel","USA"=>"United States","UK"=>"United Kingdom");
$currencies = array("NIS"=>"New Israeli Shekel","USD"=>"United States
Dollar","UK"=>"United Kingdom Pounds");
$languages = array("he"=>"Hebrew","en"=>"English","ru"=>"Russian");
var_dump($countries,$currencies,$languages);
?>
```

הפלט שנקבל הוא:

```
array(3) { ["IL"]=> string(6) "Israel" ["USA"]=> string(13) "United States"
["UK"]=> string(14) "United Kingdom" } array(3) { ["NIS"]=> string(18) "New Israeli
Shekel" ["USD"]=> string(20) "United States Dollar" ["UK"]=> string(21) "United
Kingdom Pounds" } array(3) { ["he"]=> string(6) "Hebrew" ["en"]=> string(7)
"English" ["ru"]=> string(7) "Russian" }
```

הפונקציה print_r

באמצעות פונקציה זו ניתן לשלוח להדפסה את תוכנו של composite value (מערך, לדוגמא). להבדיל מהפונקציה var_dump ניתן להשתמש בפונקציה זו כדי להדפיס מערך יחיד בלבד.

התכנית הבאה מדפיסה את תוכנו של מערך למסך באמצעות הפונקציה print_r.

```
<?php
$countries = array("IL"=>"Israel","USA"=>"United States","UK"=>"United Kingdom");
$var = print_r($countries);
?>
```

הפלט שנקבל למסך הוא:

```
Array ( [IL] => Israel [USA] => United States [UK] => United Kingdom )
```

אופן הפעולה של מערך

אופן פעולתו של מערך ב-PHP דומה לפעולתו של מבנה הנתונים map שאיבריו שומרים על הסדר שלהם כפי שנקבע בעת יצירתו. כל איבר כולל ערך (value) אשר נשמר בצירוף מפתח (key). בדרך זו ניתן להשתמש במערך כדי לדמות מבני נתונים שונים כגון queue, stack ו-map.

אפשר לדמות את אופן פעולתו של המערך כלוח מודעות עשוי משעם שכל ערך (value) מחובר למפתח (key) אשר ממוקם על לוח השעם. בדרך זו, ניתן בכל עת לקבל את כל אחד מהערכים (ה-values) באמצעות ה-key שלו.

$vec = array(“en”=>”English”, “ru”=>”Russian”,”fr”=>”French”);

הערך "English” מחובר ללוח השעם באמצעות המפתח "en”. הערך "Russian” מחובר ללוח השעם באמצעות המפתח "ru”. הערך "French” מחובר ללוח השעם באמצעות המפתח "fr”. בכל עת ניתן לקבל את כל אחד מהערכים באמצעות המפתח ששוייך לו.

$language = $vec[“en”];

כל מפתח (key) יכול לשמש ערך (value) יחיד בלבד. ניסיון להוסיף יותר מערך אחד עם אותו מפתח (key) יגרום לכך שרק הערך (ה-value) האחרון שמוסיפים אכן יתווסף.

התכנית הבאה מנסה להוסיף שני ערכים (values) שונים עם אותו מפתח (key) לאותו מערך.

```
<?php
$countries = array("UK"=>"Pound","IL"=>"Israel","USA"=>"United
States","UK"=>"United Kingdom Pound");
$var = print_r($countries);
?>
```

הפלט שנקבל מעיד על כך שבניסיון להוסיף שני ערכים (values) שונים עם אותו מפתח (key) רק הערך האחרון שמנסים להוסיף אכן מתווסף.

```
Array ( [UK] => United Kingdom Pound [IL] => Israel [USA] => United States )
```

מערך דו מימדי

מערך דו מימדי הוא מערך רגיל שכל אחד מאיבריו כולל ערך שהוא מערך אחר. כדי ליצור מערך דו מימדי יש ליצור תחילה את כל אחד מהמערכים האחרים ולהשים אותם בתור הערך של כל אחד מהאיברים שמוכנסים למערך העיקרי. התכנית הבאה מציגה את יצירתו של מערך דו מימדי רגיל.

```
<?php
$matrix = array();
$matrix[0] = array("a","b");
$matrix[1] = array("c","d");
echo $matrix[0][0];
echo $matrix[0][1];
echo $matrix[1][0];
echo $matrix[1][1];
?>
```

הפלט שנקבל הוא:

```
abcd
```

הפקודה list

באמצעות הפקודה list ניתן לבצע השמה מהירה של ערכים (values) של איברים במערך אל תוך משתנים שאנו עובדים עימם. השימוש בפקודה list נעשה באופן הבא.

יש לרשום בצד שמאל של סימן השוויון את הפקודה list ובסוגריים עגולות מיד אחריה פירוט רשימת המשתנים שלתוכם רוצים להכניס את הערכים של האיברים שמוחזקים במערך. יש להשתמש בפסיק כסימן מפריד בין המשתנים הללו. בצד ימין של סימן השוויון יש לרשום את שמו של המשתנה שמהווה את המערך שאת ערכיו להשים אל תוך המשתנים שפירטנו.

התכנית הבאה יוצרת מערך שכולל ארבעה איברים ולאחר מכן מבצעת השמה של הערכים באותם איברים אל תוך משתנים נפרדים.

```
<?php
$names=array("moshe","david","mike","ram");
list($manager,$assistant,$teacher,$president) = $names;
echo "<BR>manager=$manager";
echo "<BR>assistant=$assistant";
echo "<BR>teacher=$teacher";
echo "<BR>president=$president";
?>
```

הפלט שנקבל הוא:

```
manager=moshe
assistant=david
teacher=mike
president=ram
```

האופרטורים +, ==, ===, =! ו- ==!

באמצעות הפעלת האופרטור + על שני מערכים נקבל מערך חדש שמהווה union של שני המערכים. אופן פעולתו של אופרטור זה מתבסס על ה-keys שכל אחד משני המערכים כולל. במקרה שקיים key שמופיע גם במערך הראשון וגם במערך השני אז הערך שמופיע במערך הראשון הוא זה שיופיע במערך החדש שמקבלים.

הדוגמא הבאה כוללת הפעלה של + על שני מערכים והדפסת המערך החדש שמתקבל למסך.

```
<?php
$vecA = array(1=>"a",2=>"b",3=>"c");
$vecB = array(1=>"d",4=>"e",5=>"f",6=>"g");
$vecC = $vecA + $vecB;
print_r($vecC);
?>
```

הפלט שנקבל הוא:

```
Array ( [1] => a [2] => b [3] => c [4] => e [5] => f [6] => g )
```

השוואה בין שני מערכים באמצעות האופרטור == תחזיר true אם (ורק אם) שני המערכים מחזיקים את אותם ערכים (values) וה-keys של כל אחד מהם זהים בהתאמה. השוואה בין שני מערכים באמצעות האופרטור === מבצעת את אותה השוואה רק שבנוסף, כדי שתחזיר את הערך true יש צורך בכך שהמיקום של האיברים בכל אחד משני המערכים יהיה זהה.

האופרטורים =! ו-==! פועלים בהתאמה באופן הפוך. במקרים שבהם האופרטור == מחזיר true האופרטור =! מחזיר false ולהיפך. במקרים שבהם האופרטור === מחזיר true האופרטור ==! מחזיר false ולהיפך.

הדוגמא הבאה מציגה את אופן הפעולה של האופרטורים ==, ===, =! ו- ==!.

```
<?php
$vecA = array(1=>"a",2=>"b",3=>"c");
$vecB = array(1=>"a",2=>"b",3=>"c");
$vecC = array(3=>"c",1=>"a",2=>"b");
echo "<BR>vecA:";
print_r($vecA);
echo "<BR>vecB:";
print_r($vecB);
echo "<BR>vecC:";
print_r($vecC);
if($vecA==$vecB)
{
```

```
	echo "<BR>vecA==vecB";
}
else
{
	echo "<BR>vecA!=vecB";
}
if($vecA===$vecB)
{
	echo "<BR>vecA===vecB";
}
else
{
	echo "<BR>vecA!==vecB";
}
if($vecA==$vecC)
{
	echo "<BR>vecA==vecC";
}
else
{
	echo "<BR>vecA!=vecC";
}
if($vecA===$vecC)
{
	echo "<BR>vecA===vecC";
}
else
{
	echo "<BR>vecA!==vecC";
}
?>
```

הפלט שנקבל הוא:

```
vecA:Array ( [1] => a [2] => b [3] => c )
vecB:Array ( [1] => a [2] => b [3] => c )
vecC:Array ( [3] => c [1] => a [2] => b )
vecA==vecB
vecA===vecB
vecA==vecC
vecA!==vecC
```

הפונקציה count

באמצעות הפונקציה count ניתן לקבל את גודלו (מספר האיברים) של מערך נתון. הדוגמא הבאה מציגה את אופן הפעולה של פונקציה זו.

```
<?php
$vec = array(1=>"a",2=>"b",3=>"c");
echo count($vec);
?>
```

הפלט שנקבל הוא:

```
3
```

הפונקציה is_array

באמצעות פונקציה זו ניתן לדעת אם משתנה מסוים הוא מערך. הדוגמא הבאה מציגה הפעלה פשוטה של פונקציה זו.

```
<?php
$vec = array(1=>"a",2=>"b",3=>"c");
$num = 9;
if(is_array($vec))
{
      echo "<BR>vec is array";
}
else
{
      echo "<BR>vec is not array";
}
if(is_array($num))
{
      echo "<BR>num is array";
}
else
{
      echo "<BR>num is not array";
}
?>
```

הפלט של התכנית הוא:

```
vec is array
num is not array
```

הפונקציה isset

באמצעות הפונקציה isset ניתן לבדוק אם key מסוים כבר קיים במערך או לא.

הדוגמא הבאה מציגה את אופן השימוש בפונקציה זו.

```
<?php
$vec = array('a'=>1,'b'=>2,'c'=>3);
if(isset($vec['a'])) echo "<BR>'a' key exists";
if(isset($vec['b'])) echo "<BR>'b' key exists";
if(isset($vec['c'])) echo "<BR>'c' key exists";
if(isset($vec['d'])) echo "<BR>'d' key exists";
if(isset($vec['e'])) echo "<BR>'e' key exists";
?>
```

הפלט שנקבל הוא:

```
'a' key exists
'b' key exists
'c' key exists
```

הפונקציה array_key_exists

באמצעות פונקציה זו ניתן לדעת אם key מסוים כבר קיים במערך או לא. הדוגמא הבאה מציגה את אופן השימוש בפונקציה זו.

```
<?php
$vec = array('a'=>1,'b'=>2,'c'=>3);
if(array_key_exists('a',$vec)) echo "<BR>'a' key exists";
if(array_key_exists('b',$vec)) echo "<BR>'b' key exists";
if(array_key_exists('c',$vec)) echo "<BR>'c' key exists";
if(array_key_exists('d',$vec)) echo "<BR>'d' key exists";
if(array_key_exists('e',$vec)) echo "<BR>'e' key exists";
?>
```

הפלט שנקבל הוא:

```
'a' key exists
'b' key exists
'c' key exists
```

ההבדל בין השימוש בפונקציה זו לפונקציה isset הוא שבאותם מקרים שבהם ה-value של key מסוים הוא null הפונקציה isset תחזיר לנו null בעוד שהפונקציה array_key_exists תחזיר לנו true.

הפונקציה in_array

באמצעות פונקציה זו ניתן לבדוק אם ערך (value) מסוים קיים במערך. הדוגמא הבאה מציגה את אופן השימוש בפונקציה זו.

```
<?php
$vec = array('a','b','c','d','f','g','h');
if(in_array('a',$vec))
{
      echo "<BR>'a' exists";
}
else
{
      echo "<BR>'a' doesn't exist";
}
if(in_array('b',$vec))
{
      echo "<BR>'b' exists";
}
else
{
      echo "<BR>'b' doesn't exist";
}
if(in_array('c',$vec))
{
      echo "<BR>'c' exists";
}
else
{
      echo "<BR>'c' doesn't exist";
}
if(in_array('d',$vec))
{
      echo "<BR>'d' exists";
}
else
{
      echo "<BR>'d' doesn't exist";
```

```
}
if(in_array('e',$vec))
{
      echo "<BR>'e' exists";
}
else
{
      echo "<BR>'e' doesn't exist";
}
?>
```

הפלט שנקבל הוא:

```
'a' exists
'b' exists
'c' exists
'd' exists
'e' doesn't exist
```

הפונקציה array_flip

באמצעות פונקציה זו ניתן על בסיס מערך קיים ליצור מערך חדש שה-keys שלו הם ה-values של המערך הקיים וה-values הם ה-keys.

הדוגמא הבאה מציגה שימוש בפונקציה זו.

```
<?php
$vecA = array("en"=>"english","ru"=>"russian","he"=>"hebrew");
$vecB = array_flip($vecA);
echo "vecA:<br>";
var_dump($vecA);
echo "<br><br>vecB:<br>";
var_dump($vecB);
?>
```

הפלט שנקבל הוא:

vecA:
array(3) { ["en"]=> string(7) "english" ["ru"]=> string(7) "russian" ["he"]=> string(6) "hebrew" }

vecB:
array(3) { ["english"]=> string(2) "en" ["russian"]=> string(2) "ru" ["hebrew"]=> string(2) "he" }

הפונקציה array_reverse

באמצעות פונקציה זו ניתן על בסיס מערך קיים לקבל מערך חדש שסדר האיברים שלו הפוך לסדר האיברים במערך הקיים.

התכנית הבאה מדגימה את אופן השימוש בפונקציה זו.

```
<?php
$vecA = array("en"=>"english","ru"=>"russian","he"=>"hebrew");
$vecB = array_reverse($vecA);
echo "vecA:<br>";
var_dump($vecA);
echo "<br><br>vecB:<br>";
var_dump($vecB);
?>
```

הפלט שנקבל הוא:

vecA:
array(3) { ["en"]=> string(7) "english" ["ru"]=> string(7) "russian" ["he"]=> string(6) "hebrew" }

vecB:
array(3) { ["he"]=> string(6) "hebrew" ["ru"]=> string(7) "russian" ["en"]=> string(7) "english" }

מצביע של מערך

ניתן לעבור על הערכים שמוחזקים במערך באמצעות מצביע (pointer).

באמצעות הפונקציה reset ניתן לגרום למצביע של המערך לעבור לתחילתו.

באמצעות הפונקציה next ניתן לגרום למצביע לעבור לאיבר הבא של המערך.

באמצעות הפונקציה prev ניתן לגרום למצביע לעבור לאיבר הקודם של המערך.

באמצעות הפונקציה current ניתן לקבל את ה-value של האלמנט הנוכחי, שעליו המצביע ממוקם.

באמצעות הפונקציה key ניתן לקבל את ה-key של האלמנט הנוכחי, שעליו המצביע ממוקם.

התכנית הבאה מציגה את אופן השימוש בפונקציות אלה.

```
<?php
$vec = array("a","b","c","d","f","g","h");
reset($vec);
while(key($vec)!==null)
{
echo key($vec)." is the key and ".current($vec)." is the value<BR>";
next($vec);
}
?>
```

הפלט שנקבל הוא:

```
0 is the key and a is the value
1 is the key and b is the value
2 is the key and c is the value
3 is the key and d is the value
4 is the key and f is the value
5 is the key and g is the value
6 is the key and h is the value
```

הלולאה foreach

באמצעות הלולאה foreach ניתן לעבור על כל האיברים במערך באופן שמאפשר קבלת ה-key וה-value של כל אחד מהם.

```
foreach(____ as _____ => _______)
```

הערך הראשון (ראשון משמאל) שיש לרשום הוא שם המערך (המשתנה שמחזיק בתוכו את המערך). הערך השני (מימין למילה as הוא שם המשתנה שבכל איטרציה במעבר על פני האלמנטים של המערך יחזיק ב-key של האיבר הנוכחי והערך השלישי (מימין לאופרטור => הוא שם המשתנה שיחזיק ב-value של האיבר הנוכחי. בדרך זו ניתן לעבור על כל אחד מהאיברים שבמערך ולקבל את ה-key ואת ה-value של כל אחד מהם.

הדוגמא הבאה מציגה שימוש פשוט בלולאה foreach:

```
<?php
$vec = array("a","b","c","d","f","g","h");
foreach($vec as $key => $value)
{
      echo "<BR>key=$key value=$value";
}
?>
```

הפלט שיופיע למסך הוא:

key=0 value=a
key=1 value=b
key=2 value=c
key=3 value=d
key=4 value=f
key=5 value=g
key=6 value=h

חשוב להבין שהלולאה foreach עובדת על עותק שנוצר מהמערך המקורי. מסיבה זו, אם תוך כדי מעבר על האלמנטים באמצעות לולאת foreach ננסה לשנות את ה-value באיבר מסוים השינוי לא יתבצע במערך עצמו וכאשר הלולאה תסתיים ונבדוק את המערך נגלה שהוא כלל לא השתנה. באמצעות הוספת האופרטור & למשתנה שמחזיק את ה-value של כל איבר ואיבר ניתן לגרום לשינוי במערך שעליו הלולאה פועלת. הדוגמא הבאה מציגה זאת.

```
<?php
$vec = array(100,200,300);
echo "<br>before";
foreach($vec as $key => &$value)
{
      echo "<BR>key=$key value=$value";
      $value = $value * 10;
}
echo "<br>after";
foreach($vec as $key => $value)
{
      echo "<BR>key=$key value=$value";
}
?>
```

הפלט שיתקבל הוא:

```
before
key=0 value=100
key=1 value=200
key=2 value=300
after
key=0 value=1000
key=1 value=2000
key=2 value=2000
```

כאשר משתמשים בלולאה foreach ניתן לחילופין להתעלם מה-key ולהתייחס רק ל-value. הדוגמא הבאה מציגה זאת.

```
<?php
$vec = array(100,200,300);
foreach($vec as $value)
{
      echo "<BR>value=$value";
}
?>
```

הפלט שיהיה למסך הוא:

```
value=100
value=200
value=300
```

הפונקציה array_combine

פונקציה זו מקבלת בעת הפעלתה שני מערכים ויוצרת מערך חדש שה-keys שלו הם ה-values של המערך הראשון וה-values הם ה-values של המערך השני.

הדוגמא הבאה מציגה שימוש בסיסי בפונקציה זו.

```
<?php
$keys = array("il","us","uk");
$values = array("Israel","USA","United Kingdom");
$vec = array_combine($keys,$values);
foreach($vec as $key=>$value)
{
      echo "<BR>key=$key value=$value";
}
?>
```

הפלט שנקבל למסך הוא:

key=il value=Israel

key=us value=USA

key=uk value=United Kingdom

הפונקציה array_walk

פונקציה זו מאפשרת לעבור על כל אחד מהאיברים במערך נתון ולקרוא להפעלת פונקציה מסוימת על ה-key וה-value כל אחד מהם. הארגומנט הראשון שיש לשלוח אליה הוא המערך (שם המשתנה שמחזיק במערך) שעל האיברים שלו רוצים לעבור. הארגומנט השני שיש לשלוח הוא שם הפונקציה שאנו רוצים להפעיל על ה-key וה-value של כל אחד מהאיברים.

הפונקציה שאת שמה שולחים בתור הארגומנט השני לפונקציה array_walk תופעל על כל אחד מהאיברים. פונקציה זו צריכה להיות עם שני פרמטרים. פרמטר אחד אשר יחזיק ב-key ופרמטר שני שיחזיק ב-value.

ניתן לשלוח לפונקציה array_walk ארגומנט שלישי אשר יישלח אל הפונקציה אשר תופעל על כל אחד מהאיברים של המערך.

התכנית הבאה מדגימה את אופן השימוש בפונקציה array_walk:

```
<?php
$cars = array("T" => "Toyota", "M" => "Mazda", "S" => "Suzuki", "Y"
=> "Yamaha");
function changearray(&$val, $key, $prefix)
{
$val = "$prefix: $val";
}
function printarray($itemvalue, $itemkey)
{
echo "$itemkey : $itemvalue<br>";
}
echo "before ...<BR>";
array_walk($cars, 'printarray');
array_walk($cars, 'changearray', 'car');
echo "after...<BR>";
array_walk($cars, 'printarray');
?>
```

הפלט של התכנית הוא:

```
before ...
T : Toyota
M : Mazda
S : Suzuki
Y : Yamaha
```

```
after...
T : car: Toyota
M : car: Mazda
S : car: Suzuki
Y : car: Yamaha
```

הפונקציה array_walk_recursive

פעולתה של פונקציה זו דומה לפעולתה של הפונקציה array_walk. להבדיל מ-arraywalk פונקציה זו עוברת על כל האיברים של כל המערכים שמוחזקים כ-values של האיברים של המערך העיקרי ושל כל אחד מהמערכים האחרים שמוחזקים כ-values על ידיו באופן ישיר או עקיף.הדוגמא הבאה מציגה את אופן השימוש בפונקציה זו.

```
<?php
      $japan_cars = array(
            "T" => "Toyota",
            "M" => "Mazda",
            "S" => "Suzuki",
            "Y" => "Yamaha");
      $usa_cars = array(
            "C" => "Chevrolet",
            "P" => "Pontiac",
            "C" => "Cryzler");
      $cars = array("US" => $usa_cars, "JP" => $japan_cars);
      function changearray(&$val, $key, $prefix)
      {
            $val = "$prefix: $val";
      }
      function printarray($itemvalue, $itemkey)
      {
            echo "$itemkey : $itemvalue<br>";
      }
      echo "before ...<BR>";
      array_walk_recursive($cars, 'printarray');
      array_walk_recursive($cars, 'changearray', 'car');
      echo "after...<BR>";
      array_walk_recursive($cars, 'printarray');
?>
```

הפלט שנקבל הוא:

```
before ...
C : Cryzler
P : Pontiac
T : Toyota
M : Mazda
S : Suzuki
Y : Yamaha
after...
C : car: Cryzler
P : car: Pontiac
T : car: Toyota
M : car: Mazda
S : car: Suzuki
Y : car: Yamaha
```

מיון מערכים

שפת התיכנות PHP כוללת מגוון של פונקציות למיון מערכים. שתי הפונקציות הפשוטות ביותר הן sort ו-asort. כל אחת משתי המתודות הללו מקבלת כארגומנט (אחד לפחות) את המערך שאנו רוצים למיין (reference לאובייקט שמייצג את המערך שרוצים למיין). ניתן, בנוסף, לשלוח פרמטר שני אשר יורה על האופן שבו המיון יתבצע.

בעוד שהפונקציה asort לא משנה את ה-keys הפונקציה sort מחליפה את ה-keys במספרי אינדקס החל מ-0 (עבור ה-value הראשון) וכלה ב-index-1 (עבור ה-value האחרון).

הדוגמא הבאה מציגה שימוש בסיסי בפונקציות sort ו-asort.

```
<?php
	$vec = array("z"=>"Ziv","b"=>"Boaz","d"=>"David","a"=>"Antony");
	echo "<br>before...<br>";
	var_dump($vec);
	asort($vec);
	echo "<br>after asort()<br>";
	var_dump($vec);
	shuffle($vec);
	echo "<br>after shuffle()<br>";
	var_dump($vec);
	sort($vec);
	echo "<br>after sort()<br>";
	var_dump($vec);
?>
```

הפלט שנקבל הוא:

```
before...
array(4) { ["z"]=> string(3) "Ziv" ["b"]=> string(4) "Boaz" ["d"]=> string(5) "David" ["a"]=> string(6) "Antony" }
after sort()
array(4) { ["a"]=> string(6) "Antony" ["b"]=> string(4) "Boaz" ["d"]=> string(5) "David" ["z"]=> string(3) "Ziv" }
after shuffle()
array(4) { [0]=> string(6) "Antony" [1]=> string(3) "Ziv" [2]=> string(4) "Boaz" [3]=> string(5) "David" }
after asort()
array(4) { [0]=> string(6) "Antony" [1]=> string(4) "Boaz" [2]=> string(5) "David" [3]=> string(3) "Ziv" }
```

הארגומנט השני שניתן לשלוח לפונקציה sort ולפונקציה asort יכול להיות אחד הערכים הבאים:

SOFT_REGULAR
המיון יתבצע על פי ה-values של כל אחד מהאיברים ומבלי לבצע שום שינוי באיברים הממוינים.

SOFT_NUMERIC
תחילה, כל אחד מה-values של כל אחד מהאיברים יומר ל-numeric value ורק אחר כך המיון יתבצע (על פי אותם numeric values שהתקבלו).

SORT_STRINGS
תחילה, כל אחד מה-values של כל אחד מהאיברים יומר ל-string value ורק לאחר מכן המיון יתבצע (על פי אותם string values שהתקבלו).

הפונקציה rsort ממיינת את איברי המערך בסדר הופכי. כמו כן, היא מסירה את כל ה-keys וקובעת אחרים במקומם.

הפונקציה ksort ממיינת את איברי המערך על פי ה-keys שלהם.

הפונקציה krsort ממיינת את איברי המערך על פי ה-keys שלהם (בסדר הופכי).

הפונקציה usort ממיינת את איברי המערך על פי ה-values שלהם ותוך שימוש בפונקציה שיש לשלוח אליה בתור ארגומנט שני.
הפונקציה שיש לשלוח צריכה להיות מוגדרת עם שני פרמטרים ועליה להחזיר או 0 או ערך חיובי או ערך שלילי.

התכנית הבאה מציגה דוגמא לשימוש ב-usort.

```
<?php
	function cmp($a, $b)
	{
		if ($a == $b)
		{
			return 0;
		}
		return ($a < $b) ? -1 : 1;
	}
	$vec = array(12,532,12,56322343,232,5,2,1,1,1,4,2, 5, 6, 1);
	usort($vec, "cmp");
	foreach ($vec as $key => $value)
```

```
	{
		echo "$key: $value<BR>";
	}
?>
```

הפלט שנקבל הוא:

```
0: 1
1: 1
2: 1
3: 1
4: 2
5: 2
6: 4
7: 5
8: 5
9: 6
10: 12
11: 12
12: 232
13: 532
14: 56322343
```

הפונקציה shuffle

פונקציה זו מערבבת את איברי המערך באופן אקראי. הדוגמא הבאה מציגה שימוש בסיסי בפונקציות sort ו-asort.

```
<?php
	$vec = array("z"=>"Ziv","b"=>"Boaz","d"=>"David","a"=>"Antony");
	echo "<br>before...<br>";
	var_dump($vec);
	asort($vec);
	echo "<br>after asort()<br>";
	var_dump($vec);
	shuffle($vec);
	echo "<br>after shuffle()<br>";
	var_dump($vec);
	sort($vec);
	echo "<br>after sort()<br>";
	var_dump($vec);
?>
```

הפלט שנקבל הוא:

```
before...
array(4) { ["z"]=> string(3) "Ziv" ["b"]=> string(4) "Boaz" ["d"]=> string(5) "David" ["a"]=> string(6) "Antony" }
after sort()
array(4) { ["a"]=> string(6) "Antony" ["b"]=> string(4) "Boaz" ["d"]=> string(5) "David" ["z"]=> string(3) "Ziv" }
after shuffle()
array(4) { [0]=> string(6) "Antony" [1]=> string(3) "Ziv" [2]=> string(4) "Boaz" [3]=> string(5) "David" }
after asort()
array(4) { [0]=> string(6) "Antony" [1]=> string(4) "Boaz" [2]=> string(5) "David" [3]=> string(3) "Ziv" }
```

הפונקציה array_rand

באמצעות פונקציה זו ניתן לקבל איברים מתוך המערך אשר ייבחרו באופן רנדומלי. הארגומנט הראשון שיש לשלוח לפונקציה זו הוא המערך (שם המשתנה שמחזיק במערך שמתוכו אנו רוצים לקבל איברים באופן רנדומלי). הארגומנט השני שיש לשלוח הוא מספר האיברים שאנו רוצים לקבל באופן רנדומלי מהמערך. אם אנו מבקשים איבר רנדומלי אחד אז הפונקציה תחזיר את ה-key שלו. אם אנו מבקשים מספר איברים (גדול מ-1) אז הפונקציה תחזיר מערך אשר ערכיו הם ה-keys של אותם איברים שנבחרו באופן רנדומלי.

הדוגמא הבאה מציגה שימוש פשוט בפונקציה זו.

```
<?php
      $vec = array("a","b","c","d","f","g","h");
      $random_keys = array_rand($vec,3);
      echo $vec[$random_keys[0]];
      echo "<BR>";
      echo $vec[$random_keys[1]];
      echo "<BR>";
      echo $vec[$random_keys[2]];
      echo "<BR>";
?>
```

הפלט שנקבל בכל הרצה יהיה שונה. כך, למשל, הוא עשוי להיות הפלט הבא:

```
b
c
f
```

שימוש במערך כ Stack

באמצעות הפונקציות array_push ו-array_pop ניתן להשתמש במערך כמחסנית.

הפונקציה array_push מקבלת שני ארגומנטים. הארגומנט הראשון הוא המערך (השם של המשתנה שמחזיק במערך) והארגומנט השני הוא הערך שאנו רוצים להוסיף לו (לסופו). בכל הוספה של ערך נוסף גודל המערך גדל ב-1. ניתן לשלוח ארגומנטים נוספים במידה שרוצים להוסיף מספר ערכים חדשים בו זמנית. הפונקציה מחזירה את מספר האיברים שיש במערך לאחר ההוספה.

הפונקציה array_pop מחזירה את הערך האחרון שהתווסף למערך ומקטינה את אורכו ב-1.

הדוגמא הבאה מציגה שימוש פשוט בפונקציות אלה.

```
<?php
	$vec = array();
	array_push($vec,"a");
	array_push($vec,"shalom");
	$num = array_push($vec,"israel");
	echo "<br>current num of elements is ".$num;
	$str = array_pop($vec);
	echo "<br>last added element is ".$str;
?>
```

הפלט שנקבל הוא:

```
current num of elements is 3
last added element is israel
```

שימוש במערך כ Queue

באמצעות הפונקציות array_shift ו-array_unshift ניתן להשתמש במערך בדומה למבנה הנתונים תור.

הפונקציה array_shift מחזירה את האיבר הראשון של המערך, מוזיזה את כל האיברים שמאלה ובמידה שה-keys הם מספרים אז הפונקציה תעדכן אותם כך שמספר האינדקס של האלמנט הראשון יתחיל מאפס.

הפונקציה array_unshift מקבלת איבר (או איברים) שאנו רוצים להוסיף למערך ומוסיפה אותו (אותם) לתחילתו. הפונקציה מוזיזה את כל האיברים של המערך ימינה ובמידה שה-keys הם מספרי אינדקס אז היא מעדכנת אותם כך שמספר האינדקס של האיבר הראשון שהתווסף יהיה 0.

הדוגמא הבאה מציגה שימוש בפונקציות אלה.

```
<?php
	$vec = array();
	array_unshift($vec,"salam");
	array_unshift($vec,"shalom");
	array_unshift($vec,"peace");
	echo "<br>before...";
	var_dump($vec);
	echo "<br>";
	$ob = array_shift($vec);
	echo "<br>after...";
	var_dump($vec);
?>
```

הפלט שנקבל הוא:

```
before...
array(3)
{ [0]=> string(5) "peace" [1]=> string(6) "shalom" [2]=> string(5) "salam" }

after...
array(2)
{ [0]=> string(6) "shalom" [1]=> string(5) "salam" }
```

שימוש במערך כ Set

כאשר מפעילים את פונקציה array_diff יש לשלוח אליה שני מערכים. הפונקציה מחזירה מערך חדש אשר מכיל את כל האיברים שמופיעים במערך הראשון ושלא מופיעים בשני. הפונקציה מתבססת בפעולתה על הייצוג המחרוזתי של כל אחד מה-values.

הדוגמא הבאה מציגה שימוש בסיסי בפונקציה זו.

```
<?php
	$vecA = array("a","b","c");
	$vecB = array("a","d","e");
	$vecC = array_diff($vecA,$vecB);
	echo "<br>vecA...";
	var_dump($vecA);
	echo "<br>vecB...";
	var_dump($vecB);
	echo "<br>vecC...";
	var_dump($vecC);
?>
```

הפלט שנקבל למסך הוא.

```
vecA...array(3) { [0]=> string(1) "a" [1]=> string(1) "b" [2]=> string(1) "c" }
vecB...array(3) { [0]=> string(1) "a" [1]=> string(1) "d" [2]=> string(1) "e" }
vecC...array(2) { [1]=> string(1) "b" [2]=> string(1) "c" }
```

באמצעות הפונקציה array_diff_assoc מקבלים פונקציונליות דומה בהבדל אחד. במקום להתבסס על הייצוג המחרוזתי של כל אחד מהאלמנטים היא משווה גם את ה-keys וגם את ה-values.

באמצעות הפונקציה array_diff_ukey מקבלים פונקציונליות דומה בהבדל אחד. היא מתבססת על ה-keys בלבד.

הפונקציה array_intersect מאפשרת לנו לקבל את כל ה-values שמופיעים גם במערך הראשון גם במערך השני. הפונקציה מקבלת שני ארגומנטים (את שני המערכים הנתונים) ומחזירה מערך חדש אשר כולל את כל האיברים שמופיעים גם במערך הראשון וגם במערך השני.

התכנית הבאה מציגה שימוש בסיסי בפונקציה array_intersect.

```
<?php
	$vecA = array("a","b","c","e","f");
	$vecB = array("a","d","e","z","m");
	$vecC = array_intersect($vecA,$vecB);
	echo "<br>vecA...";
	var_dump($vecA);
	echo "<br>vecB...";
	var_dump($vecB);
	echo "<br>vecC...";
	var_dump($vecC);
?>
```

הפלט שנקבל הוא:

```
vecA...array(5) { [0]=> string(1) "a" [1]=> string(1) "b" [2]=> string(1) "c" [3]=>
string(1) "e" [4]=> string(1) "f" }

vecB...array(5) { [0]=> string(1) "a" [1]=> string(1) "d" [2]=> string(1) "e" [3]=>
string(1) "z" [4]=> string(1) "m" }

vecC...array(2) { [0]=> string(1) "a" [3]=> string(1) "e" }
```