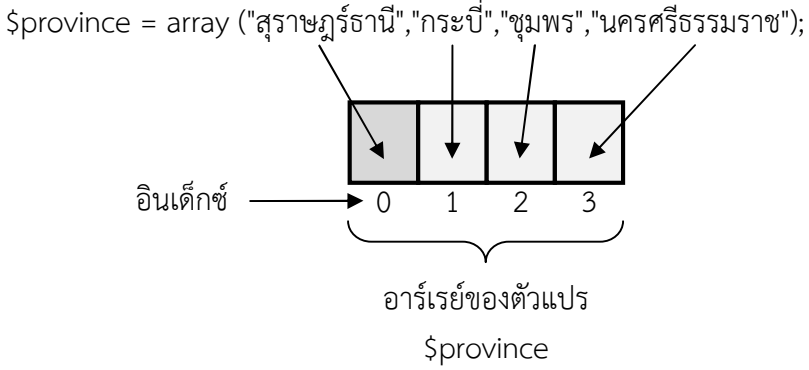
บทที่ 6
อาร์เรย์

อาร์เรย์ คือ ชุดของข้อมูล (Data Sets) หรือชุดของตัวแปร ใช้สำหรับเก็บค่าของข้อมูลที่อยู่ในกลุ่มเดียวกัน มีการเรียงลำดับที่แน่นอน ลำดับของอาร์เรย์โดยปกติจะเริ่มจากศูนย์ และเป็นลำดับต่อเนื่องไปจนถึงตัวสุดท้าย อาร์เรย์แตกต่างจากตัวแปรทั่วไป คือ ตัวแปรโดยทั่วไปจะถูกจัดเก็บในหน่วยความจำแบบไม่ต่อเนื่องกัน แต่ตัวแปรประเภทอาร์เรย์จะถูกเก็บในหน่วยความจำในตำแหน่งที่ต่อเนื่องกัน

ข้อมูลแต่ละตัวในอาร์เรย์เรียกว่า "สมาชิก (Member) หรืออิลิเมนต์ (Element)" โดยสมาชิกเหล่านี้มักจะมีความสัมพันธ์กันในลักษณะใดลักษณะหนึ่ง สำหรับใน PHP ข้อมูลที่เก็บในอาร์เรย์ไม่จำเป็นต้องเป็นข้อมูลชนิดเดียวกัน สมาชิกแต่ละตัวจะประกอบด้วยค่าข้อมูล (Value) และอินเด็กซ์ (Index) หรือคีย์ (Key) ของอาร์เรย์ที่ใช้อ้างอิงถึงตำแหน่งของสมาชิกแต่ละตัวในอาร์เรย์ ตัวอย่างรูปแบบการจัดเก็บตัวแปรอาร์เรย์ในหน่วยความจำ ดังภาพที่ 6.1



ภาพที่ 6.1 ตัวอย่างการจัดเก็บตัวแปรอาร์เรย์ $province ในหน่วยความจำ

จากภาพที่ 6.1 แสดงตัวอย่างการจัดเก็บตัวแปรอาร์เรย์ $province ในหน่วยความจำ เมื่อกำหนดด้วยคำสั่ง $province = array ("สุราษฎร์ธานี","กระบี่","ชุมพร","นครศรีธรรมราช"); ค่าของอินเด็กซ์จะเริ่มจาก 0 ถึง 3 และจะมีค่าเท่ากับข้อมูลที่กำหนดให้เรียงตามลำดับ ดังตารางที่ 6.1 แสดงความสัมพันธ์ระหว่างอินเด็กซ์กับค่าของข้อมูลที่กำหนดให้ตัวแปร $province

ตารางที่ 6.1 แสดงความสัมพันธ์ระหว่างอินเด็กซ์กับค่าของข้อมูลที่กำหนดให้ตัวแปร $province

ลำดับที่ของสมาชิก	อินเด็กซ์	ค่าของข้อมูล
1	$province [0]	สุราษฎร์ธานี
2	$province [1]	กระบี่
3	$province [2]	ชุมพร
4	$province [3]	นครศรีธรรมราช

6.1 การสร้างอาร์เรย์

การสร้างอาร์เรย์หลักๆ มี 4 วิธี คือ 1) การใช้ฟังก์ชัน array ( ) 2) การใช้ฟังก์ชัน range ( ) 3) การสร้างโดยใช้เครื่องหมายวงเล็บก้ามปู ([ ]) และ 4) การสร้างโดยนำค่ามาจากเท็กซ์ไฟล์ มีตัวอย่าง ดังนี้

6.1.1 การสร้างอาร์เรย์โดยใช้ฟังก์ชัน array ( )

1) การสร้างอาร์เรย์โดยใช้ฟังก์ชัน array ( ) แบบไม่กำหนดอินเด็กซ์ให้กับสมาชิก ดังนี้

รูปแบบ

```
$ชื่ออาร์เรย์ = array (สมาชิกลำดับที่ 1, สมาชิกลำดับที่ 2, ... , สมาชิกลำดับที่ N);
```

ตัวอย่างที่ 6.1 การสร้างอาร์เรย์โดยใช้ฟังก์ชัน array ( ) ไม่กำหนดอินเด็กซ์ให้กับสมาชิก

```
1   <?php
2       $province = array ("สุราษฎร์ธานี", "กระบี่", "ชุมพร", "นครศรีธรรมราช");
3       print_r ($province);        // แสดงข้อมูลในอาร์เรย์
4   ?>
```

ตัวอย่างที่ 6.2 การสร้างอาร์เรย์โดยใช้ฟังก์ชัน array ( ) แบบไม่กำหนดอินเด็กซ์ร่วมกับวงเล็บก้ามปูเพื่อกำหนดค่าให้กับสมาชิกในอาร์เรย์

```
1   <?php
2       $province = array ( );
3       $province [ ] = "สุราษฎร์ธานี";
4       $province [ ] = "กระบี่";
5       $province [ ] = "ชุมพร";
6       $province [ ] = "นครศรีธรรมราช";
7       print_r ($province);
8   ?>
```

ผลลัพธ์ของตัวอย่างที่ 6.1 และ 6.2 เหมือนกัน ดังนี้

```
Array ( [0] => สุราษฎร์ธานี [1] => กระบี่ [2] => ชุมพร [3] => นครศรีธรรมราช )
```

การอ้างถึงสมาชิกแต่ละตัวต้องระบุหมายเลขอินเด็กซ์ เริ่มต้นจากเลข 0 และ 1, 2, 3 ตามลำดับ การระบุเลขอินเด็กซ์จะต้องใส่ภายในเครื่องหมายวงเล็บก้ามปู เช่น echo $province [0] หมายถึง แสดงอาร์เรย์อินเด็กซ์ที่ 0 คือ สุราษฎร์ธานี เป็นต้น

หมายเหตุ

ฟังก์ชันที่ใช้แสดงผลข้อมูลชนิดอาร์เรย์ มีให้เลือกใช้ดังนี้

1) ฟังก์ชัน print_r ( ) ใช้แสดงข้อมูลจากตัวแปรทุกชนิด ไม่ว่าจะเป็น string, integer, array, object มีรูปแบบคำสั่ง ดังนี้

mixed print_r (mixed $expression [, bool $return = false])

เมื่อ $expression หมายถึง ชื่อตัวแปรที่ต้องการนำข้อมูลมาแสดง

$return หมายถึง กำหนดให้มีการส่งค่าคืนหรือไม่ โดยปกติจะเป็น false คือ ไม่มีการส่งค่าคืน และแสดงข้อมูลออกทางหน้าจอ หากกำหนดเป็น true จะส่งคืนค่าและต้องกำหนดตัวแปรมารับค่า และไม่แสดงข้อมูลออกทางหน้าจอ

2) ฟังก์ชัน var_dump ( ) ใช้แสดงข้อมูลเช่นเดียวกับ print_r ( ) แต่สามารถใช้กับอาร์เรย์หลายมิติได้ และไม่มีการส่งคืนค่า รูปแบบดังนี้

void var_dump (mixed $expression [, mixed $...])

2) การสร้างอาร์เรย์โดยใช้ฟังก์ชัน array ( ) แบบกำหนดคีย์ให้กับสมาชิก มีรูปแบบดังนี้

```
$ชื่ออาร์เรย์ = array (คีย์ 1=>สมาชิกลำดับที่ 1, คีย์ 2=>สมาชิกลำดับที่ 2, ... );
```

ตัวอย่างที่ 6.3 การสร้างอาร์เรย์โดยใช้ฟังก์ชัน array ( ) แบบกำหนดคีย์ให้กับสมาชิก

```
1   <?php
2       $province = array ("Surat" => "สุราษฎร์ธานี", "Krabi" => "กระบี่", "Chumphon" =>
"ชุมพร", "Nakhon" => "นครศรีธรรมราช");
3       print_r ($province);
4   ?>
```

ผลลัพธ์

```
Array ( [Surat] => สุราษฎร์ธานี [Krabi] => กระบี่ [Chumphon] => ชุมพร [Nakhon] =>
นครศรีธรรมราช )
```

การอ้างถึงสมาชิกแต่ละตัวต้องระบุคีย์ การระบุคีย์จะต้องใส่ภายในเครื่องหมายวงเล็บก้ามปู เช่น echo $province [Surat] หมายถึง แสดงข้อมูลสมาชิกคีย์ Surat คือ สุราษฎร์ธานี เป็นต้น

ตัวอย่างที่ 6.4 การสร้างอาร์เรย์โดยใช้ฟังก์ชัน array ( ) กำหนดอินเด็กซ์ให้กับสมาชิก

```
1   <?php
2       $province = array ( );                  // สร้างอาร์เรย์ว่าง
3       $province [0] = "สุราษฎร์ธานี";        // กำหนดสมาชิก
```

```
4           ...
5           $province [0] = "ยะลา";                    // แก้ไขปรับปรุงค่าข้อมูลสมาชิกเดิม
6           $province [7] = "เชียงใหม่";               // เพิ่มข้อมูลสมาชิกใหม่
7           print_r ($province);
8       ?>
```

จากตัวอย่างที่ 6.4 การสร้างอาร์เรย์โดยใช้ฟังก์ชัน array ( ) กำหนดอินเด็กซ์ให้กับสมาชิก จะเห็นได้ว่าตำแหน่งบรรทัดที่ 3 มีการกำหนดให้ตัวแปร $province [0] = "สุราษฎร์ธานี" และกำหนดค่าใหม่ให้กับตัวแปร $province [0] = "ยะลา" ซึ่งประโยชน์ของการกำหนดอินเด็กซ์ให้กับสมาชิกจะทำให้ทราบที่อยู่ที่แน่นอนของสมาชิกแต่ละตัว ทำให้สะดวกในการเปลี่ยนแปลงข้อมูลหรืออื่นๆ

6.1.2 การสร้างอาร์เรย์โดยใช้ฟังก์ชัน range ( )

ฟังก์ชัน range ( ) เป็นฟังก์ชันที่ใช้สำหรับสร้างอาร์เรย์ โดยกำหนดค่าข้อมูลเป็นช่วงของตัวเลขหรือตัวอักษร เรียงลำดับจากน้อยไปหามากหรือมากไปหาน้อย (เป็นลักษณะข้อมูลที่เกี่ยวเนื่องกันเป็นชุด) มีรูปแบบ ดังนี้

รูปแบบ

```
array range (mixed $start, mixed $limit [, number $step = 1])
```

เมื่อ $start หมายถึง ค่าเริ่มต้นของลำดับ

$limit หมายถึง ค่าสุดท้ายของลำดับ

$step หมายถึง ค่าความต่างของข้อมูล จะกำหนดหรือไม่ก็ได้ ถ้าไม่กำหนดค่าจะเพิ่มครั้งละ 1 ค่า

ตัวอย่างที่ 6.5 การสร้างอาร์เรย์โดยใช้ฟังก์ชัน range ( )

```
1       <?php
2           foreach (range (0, 12) as $number) echo $number . " ";
3           foreach (range (0, 100, 10) as $number) echo $number . " ";
4           foreach (range ('a', 'i') as $character) echo $character . " ";
5           foreach (range ('i', 'a') as $number) echo $number . " ";
6       ?>
```

จากตัวอย่างที่ 6.5 การใช้ฟังก์ชัน range ( ) อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $number เป็นช่วงของเลข โดยใช้ฟังก์ชัน range ( ) กำหนดพารามิเตอร์ค่าเริ่มต้นของลำดับ คือ 0 และค่าสุดท้าย คือ 12 ร่วมกับโครงสร้างเงื่อนไขทำซ้ำ foreach ผลลัพธ์ที่แสดงผล คือ 0 1 2 3 4 5 6 7 8 9 10 11 12

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $number เป็นช่วงของเลข โดยใช้ฟังก์ชัน range ( ) กำหนดพารามิเตอร์ค่าเริ่มต้นของลำดับ คือ 0 และค่าสุดท้าย คือ 100 ค่าความต่างของข้อมูล คือ 10 ร่วมกับโครงสร้างเงื่อนไขทำซ้ำ foreach ผลลัพธ์ที่แสดงผล คือ 0 10 20 30 40 50 60 70 80 90 100

บรรทัดที่ 4 กำหนดค่าให้กับตัวแปร $character เป็นช่วงของตัวอักษร โดยใช้ฟังก์ชัน range ( ) กำหนดพารามิเตอร์ค่าเริ่มต้นของลำดับ คือ 'a' และค่าสุดท้าย คือ 'i' ร่วมกับโครงสร้างเงื่อนไขทำซ้ำ foreach ผลลัพธ์ที่แสดงผล คือ a b c d e f g h i

บรรทัดที่ 5 กำหนดค่าให้กับตัวแปร $character เป็นช่วงของตัวอักษร โดยใช้ฟังก์ชัน range ( ) กำหนดพารามิเตอร์ค่าเริ่มต้นของลำดับ คือ 'i' และค่าสุดท้าย คือ 'a' ร่วมกับโครงสร้างเงื่อนไขทำซ้ำ foreach ผลลัพธ์ที่แสดงผล คือ i h g f e d c b a

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

6.1.3 การสร้างอาร์เรย์โดยใช้เครื่องหมายวงเล็บก้ามปู

การสร้างอาร์เรย์ด้วยวิธีนี้ เป็นการกำหนดค่าให้กับอาร์เรย์โดยตรง โดยสร้างครั้งละ 1 สมาชิก นอกจากนี้ยังสามารถนำไปใช้ในการเพิ่มสมาชิกให้กับอาร์เรย์ที่มีอยู่แล้วก็ได้

รูปแบบ

```
array array_name [Key] = value;
```

เมื่อ array_name หมายถึง ชื่อตัวแปรอาร์เรย์

Key หมายถึง อินเด็กซ์ของตัวแปรอาร์เรย์ (จะกำหนดหรือไม่ก็ได้)

value หมายถึง ค่าของข้อมูลที่จะกำหนดให้ตัวแปรอาร์เรย์

ตัวอย่างที่ 6.6 การสร้างอาร์เรย์โดยใช้เครื่องหมายวงเล็บก้ามปู [ ]

```
1   <?php
2       $country [1] = "Thailand";
3       $country [A] = "Japan";
4       $country [ ] = "Cambodia";
5       print_r ($country);
6   ?>
```

ผลลัพธ์

```
Array ( [1] => Thailand [A] => Japan [2] => Cambodia )
```

จากตัวอย่างที่ 6.6 การใช้ฟังก์ชัน range ( ) อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $country อินเด็กซ์ 1 มีค่าเท่ากับ "Thailand"

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $country คีย์ A มีค่าเท่ากับ "Japan"

บรรทัดที่ 4 กำหนดค่าให้กับตัวแปร $country แบบไม่กำหนดอินเด็กซ์หรือคีย์ มีค่าเท่ากับ "Cambodia" (การเพิ่มสมาชิกในอาร์เรย์แบบไม่กำหนดอินเด็กซ์หรือคีย์ ต่อจากสมาชิกเดิมที่มีอยู่ก่อนหน้านั้น จะตรวจสอบหมายเลขอินเด็กซ์ลำดับก่อนหน้ามีการระบุหมายเลขอะไร แล้วเพิ่มค่าอีก 1 มากำหนดเป็นอินเด็กซ์ให้กับสมาชิกใหม่)

บรรทัดที่ 5 แสดงผลข้อมูลอาร์เรย์ ที่จัดเก็บในตัวแปร $country

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

6.1.4 การสร้างอาร์เรย์โดยนำค่ามาจากเท็กซ์ไฟล์

การสร้างอาร์เรย์โดยนำค่ามาจากเท็กซ์ไฟล์เป็นวิธีการดึงข้อมูลจากเท็กซ์ไฟล์มาเก็บไว้ในตัวแปรชนิดอาร์เรย์ ซึ่งข้อมูลแต่ละบรรทัดในเท็กซ์ไฟล์ ก็คือ ค่าข้อมูลของสมาชิกแต่ละตัวในอาร์เรย์

ตัวอย่าง ข้อมูลในเท็กซ์ไฟล์ กำหนดชื่อ text.txt

```
Suratthani
Chumphon
Ranong
```

ตัวอย่างที่ 6.7 การสร้างอาร์เรย์โดยนำค่ามาจากเท็กซ์ไฟล์

```
1   <?php
2       $txt_file = file ("text.txt");
3       $count_arr = count ($txt_file);
4       if ($count_arr == 0) {
5           echo "ไม่มีข้อมูลในไฟล์ <br>";
6       } else {
7           print_r($txt_file);
8       }
9   ?>
```

ผลลัพธ์

```
Array ( [0] => Suratthani [1] => Chumphon [2] => Ranong )
```

จากตัวอย่างที่ 6.7 การสร้างอาร์เรย์โดยนำค่ามาจากเท็กซ์ไฟล์ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 เรียกใช้ฟังก์ชัน file ( ) เพื่ออ่านค่าข้อมูลทั้งหมดจากเท็กซ์ไฟล์ที่ระบุ แล้วนำมากำหนดค่าให้กับตัวแปร $txt_file (เป็นข้อมูลชนิดอาร์เรย์โดยปริยาย)

บรรทัดที่ 3 เรียกใช้ฟังก์ชัน count ( ) เพื่อนับจำนวนสมาชิกของอาร์เรย์ (ข้อมูลเท็กซ์ไฟล์ 1 บรรทัด จะมีค่าเท่ากับ 1 สมาชิกอาร์เรย์) แล้วนำผลของการนับจำนวนสมาชิกมากำหนดให้กับตัวแปร $count_txt

บรรทัดที่ 4 ใช้โครงสร้างเงื่อนไข if ตรวจสอบค่าของตัวแปร $count_arr มีค่าเท่ากับ 0 จริงหรือไม่ (หากมีค่าเท่ากับ 0 แสดงว่าไม่มีข้อมูล) ถ้าเป็นจริงไปบรรทัดที่ 5 ถ้าเป็นเท็จไปบรรทัดที่ 6

บรรทัดที่ 5 แสดงข้อความ ไม่มีข้อมูลในไฟล์

บรรทัดที่ 6 ปิดโครงสร้างเงื่อนไข if เข้าสู่โครงสร้างเงื่อนไข else (กรณีจำนวนอาร์เรย์มากกว่า 0 หมายถึง มีข้อมูลที่สามารถอ่านได้จากเท็กซ์ไฟล์)

บรรทัดที่ 7 ใช้ฟังก์ชัน print_f เพื่อแสดงผลข้อมูลชนิดอาร์เรย์ $txt_file

บรรทัดที่ 8 ปิดโครงสร้างเงื่อนไข else

บรรทัดที่ 9 สิ้นสุดสคริปต์ PHP

6.2 การเข้าถึงข้อมูลภายในอาร์เรย์

การเข้าถึงข้อมูลภายในอาร์เรย์มี 4 วิธี คือ 1) การอ้างตำแหน่งของอินเด็กซ์หรือคีย์ 2) การใช้โครงสร้างเงื่อนไขทำซ้ำ 3) การใช้ฟังก์ชัน each ( ) และ 4) การใช้ฟังก์ชัน list ( ) มีรายละเอียดดังนี้

6.2.1 การอ้างตำแหน่งของอินเด็กซ์หรือคีย์

ตัวแปรทั่วไปจะใช้เฉพาะชื่อของตัวแปรเพื่ออ้างถึงค่าในตัวแปร แต่สำหรับอาร์เรย์จะใช้ชื่อของอาร์เรย์และอินเด็กซ์หรือคีย์ในการอ้างถึงค่าในอาร์เรย์ที่ต้องการนำไปประมวลผล ค่าของอินเด็กซ์หรือคีย์ที่ต้องการอ้างถึงจะกำหนดในเครื่องหมายวงเล็บก้ามปู ซึ่งอยู่หลังชื่อตัวแปรอาร์เรย์ ตัวอย่างเช่น $province [0] เป็นการอ้างถึงสมาชิกของอาร์เรย์อินเด็กซ์ลำดับที่ 0 เป็นต้น นอกจากนี้ค่าที่อยู่ในอาร์เรย์ยังสามารถเปลี่ยนแปลงได้โดยใช้รูปแบบกำหนดค่า (เครื่องหมาย =) เหมือนการกำหนดค่าให้กับตัวแปรทั่วไป

6.2.2 การใช้โครงสร้างเงื่อนไขทำซ้ำ

การเข้าถึงข้อมูลในอาร์เรย์ด้วยการใช้โครงสร้างเงื่อนไขทำซ้ำ เช่น for, while และ foreach เป็นต้น จะใช้ได้เฉพาะอาร์เรย์ที่มีอินเด็กซ์ที่เป็นตัวเลขที่มีการเรียงลำดับแล้ว ตัวอย่างการใช้งาน ดังนี้

ตัวอย่างที่ 6.8 การใช้โครงสร้างเงื่อนไขทำซ้ำเพื่อควบคุมการแสดงผลของอาร์เรย์

```
1    <?php
2        $province = array ("สุราษฎร์ธานี", "กระบี่", "ชุมพร", "นครศรีธรรมราช");
3        $count_province = count ($province);
4        for ($loop = 0; $loop < $count_province; $loop++) {
```

```
5                   printf ("ลำดับที่ %d คือ %s <br>", $loop+1, $province [$loop]);
6           }
7       ?>
```

ผลลัพธ์

```
ลำดับที่ 1 คือ สุราษฎร์ธานี
ลำดับที่ 2 คือ กระบี่
ลำดับที่ 3 คือ ชุมพร
ลำดับที่ 4 คือ นครศรีธรรมราช
```

จากตัวอย่างที่ 6.8 การใช้โครงสร้างเงื่อนไขทำซ้ำ for เพื่อควบคุมการแสดงผลของอาร์เรย์ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $province เป็นตัวแปรชนิดอาร์เรย์ ประกอบด้วย "สุราษฎร์ธานี", "กระบี่", "ชุมพร", "นครศรีธรรมราช" ตามลำดับ

บรรทัดที่ 3 ใช้ฟังก์ชัน count ( ) นับจำนวนสมาชิกอาร์เรย์ในตัวแปร $province แล้วกำหนดค่าผลจำนวนสมาชิกให้กับตัวแปร $count_province

บรรทัดที่ 4 ใช้โครงสร้างเงื่อนไขทำซ้ำ for วนรอบเพื่ออ่านและแสดงค่าข้อมูลสมาชิกอาร์เรย์แต่ละตัวตามลำดับ

บรรทัดที่ 5 แสดงค่าข้อมูลสมาชิกอาร์เรย์แต่ละตัวตามลำดับ

บรรทัดที่ 6 ปิดโครงสร้างเงื่อนไขทำซ้ำ for

บรรทัดที่ 7 สิ้นสุดสคริปต์ PHP

6.2.3 การใช้ฟังก์ชัน each ( )

ฟังก์ชัน each ( ) เป็นฟังก์ชันที่ใช้อ่านค่าอาร์เรย์ โดยจะส่งคืนค่าเป็นคีย์ และค่าข้อมูล หากอ่านค่าในอาร์เรย์จนหมดแล้ว จะส่งคืนค่าเป็น false

รูปแบบ

```
array each ( array &$array )
```

เมื่อ $array หมายถึง ตัวแปรอาร์เรย์ที่ต้องการอ่านค่า

ตัวอย่างที่ 6.9 การใช้ฟังก์ชัน each ( ) สำหรับอ่านค่าอาร์เรย์

```
1       <?php
2           $province = array ("Surat" => "สุราษฎร์ธานี", "Krabi" => "กระบี่", "Chumphon" =>
"ชุมพร", "Nakhon" => "นครศรีธรรมราช");
3           while ($element = each ($province)) {
```

```
4                printf ("คีย์ที่ %s มีค่าเท่ากับ %s <br>",$element ["key"], $element ["value"]);
5            }
6        ?>
```

ผลลัพธ์

```
คีย์ Surat มีค่าเท่ากับ สุราษฎร์ธานี
คีย์ Krabi มีค่าเท่ากับ กระบี่
คีย์ Chumphon มีค่าเท่ากับ ชุมพร
คีย์ Nakhon มีค่าเท่ากับ นครศรีธรรมราช
```

6.2.4 การใช้ฟังก์ชัน list ( )

ฟังก์ชัน list ( ) ทำงานในลักษณะเดียวกับฟังก์ชัน each ( ) แตกต่างกันที่ฟังก์ชัน list ( ) ไม่มีการส่งคืนค่า และจะแยกอาร์เรย์ออกจากกัน แล้วเก็บค่าของสมาชิกแต่ละตัวไว้ในตัวแปรที่กำหนดไว้

รูปแบบ

```
array array_values ( array $input )
```

เมื่อ $input หมายถึง ตัวแปรอาร์เรย์ที่ต้องการอ่านค่า

ตัวอย่างที่ 6.10 การใช้ฟังก์ชัน list ( ) สำหรับอ่านค่าอาร์เรย์

```
1    <?php
2            $data_set = array ("A1001", "ปริญญา", "น้อยดอนไพร", "อาจารย์", "42000");
3            list ($code, $name, $lastname, $occupation, $salary) = $data_set;
4            echo "รหัสพนักงาน: $code ";
5            echo "ชื่อ-สกุล: $name $lastname ";
6            echo "อาชีพ: $occupation เงินเดือน: $salary";
7    ?>
```

ผลลัพธ์

```
รหัสพนักงาน: A1001
ชื่อ-สกุล: ปริญญา น้อยดอนไพร
อาชีพ: อาจารย์ เงินเดือน: 42000
```

6.3 ฟังก์ชันอื่นๆ ที่เกี่ยวกับอาร์เรย์

สำหรับใน PHP จะมีฟังก์ชันที่เกี่ยวข้องกับอาร์เรย์อยู่มากมาย แต่บางฟังก์ชันก็อาจไม่ค่อยได้ใช้งานบ่อยนัก หรืออาจใช้ฟังก์ชันอื่นทดแทนกันได้ ดังนั้นในส่วนนี้จะเน้นเฉพาะฟังก์ชันหลักๆ ที่สำคัญมักจะ

ใช้บ่อยครั้ง ทั้งนี้เพื่อความสะดวกต่อการทำความเข้าใจ จะแยกอธิบายตามความเกี่ยวข้องของฟังก์ชันต่างๆ ดังนี้

6.3.1 ฟังก์ชันในการอ่านค่าและคีย์ข้อมูลชนิดอาร์เรย์

1) การใช้ฟังก์ชัน array_keys ( )

ฟังก์ชัน array_keys ( ) ใช้สำหรับอ่านค่าคีย์ทั้งหมดของตัวแปรอาร์เรย์ โดยผลลัพธ์จะเป็นอาร์เรย์ของคีย์ที่อ่านได้ทั้งหมด

รูปแบบ

```
array array_keys ( array $input [, mixed $search_value = NULL [, bool $strict = false ]] )
```

เมื่อ $input หมายถึง ตัวแปรอาร์เรย์ที่ต้องการอ่านค่าคีย์

$search_value หมายถึง อ่านค่าคีย์เฉพาะค่าของอาร์เรย์ที่ตรงกับค่าที่กำหนด (จะกำหนดหรือไม่ก็ได้)

$strict หมายถึง ใช้สำหรับการเปรียบเทียบ (===) ใช้ร่วมกับ $search_value เพื่อเปรียบเทียบค่าของอาร์เรย์กับค่าที่เปรียบเทียบ

ตัวอย่างที่ 6.11 การใช้ฟังก์ชัน array_keys ( ) สำหรับอ่านค่าคีย์ทั้งหมดของตัวแปรอาร์เรย์

```
1   <?php
2       $array = array (0 => 100, "color" => "red");
3       print_r (array_keys ($array)); echo "<br>";
4       $array = array ("blue", "red", "green", "blue", "blue");
5       print_r (array_keys ($array, "blue")); echo "<br>";
6   ?>
```

ผลลัพธ์

```
Array ( [0] => 0 [1] => color )
Array ( [0] => 0 [1] => 3 [2] => 4 )
```

จากตัวอย่างที่ 6.11 การใช้ฟังก์ชัน array_keys ( ) สำหรับอ่านค่าคีย์ทั้งหมดของตัวแปรอาร์เรย์ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $array เป็นตัวแปรชนิดอาร์เรย์ มีค่าตามลำดับดังนี้ $array [0] มีค่าเท่ากับ 100 และ $array ["color"] มีค่าเท่ากับ "red"

บรรทัดที่ 3 แสดงผลลัพธ์เป็นอาร์เรย์ของคีย์ที่อ่านได้ทั้งหมด ของตัวแปร $array คือ Array ([0] => 0 [1] => color) หมายความว่า คีย์ที่อ่านได้ ประกอบด้วย 0 และ color

บรรทัดที่ 4 กำหนดให้ตัวแปร $array เป็นตัวแปรชนิดอาร์เรย์ มีค่าตามลำดับดังนี้

$array [0] มีค่าเท่ากับ "blue"

$array [1] มีค่าเท่ากับ "red"

$array [2] มีค่าเท่ากับ "green"

$array [3] มีค่าเท่ากับ "blue"

$array [4] มีค่าเท่ากับ " blue "

บรรทัดที่ 5 แสดงผลลัพธ์เป็นอาร์เรย์ของคีย์ที่ตรงกับคำว่า "blue" ของตัวแปรอาร์เรย์ $array ผลลัพธ์ที่แสดง คือ Array ([0] => 0 [1] => 3 [2] => 4) หมายความว่า คีย์ที่ตรงกับคำว่า "blue" อยู่ในลำดับที่ 0, 3 และ 4

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

2) การใช้ฟังก์ชัน array_values ( )

ฟังก์ชัน array_values ( ) ใช้สำหรับอ่านค่าของสมาชิกในอาร์เรย์ โดยผลลัพธ์จะเป็นอาร์เรย์ของค่าที่อ่านได้ทั้งหมด

รูปแบบ

```
array array_values ( array $input )
```

เมื่อ $input หมายถึง ตัวแปรอาร์เรย์ที่ต้องการอ่านค่า

ตัวอย่างที่ 6.12 การใช้ฟังก์ชัน array_values ( ) สำหรับอ่านค่าของสมาชิกในอาร์เรย์

```
1   <?php
2       $array = array ("size" => "XL", "color" => "gold");
3       print_r (array_values ($array));
4   ?>
```

ผลลัพธ์

```
Array ( [0] => XL [1] => gold )
```

จากตัวอย่างที่ 6.12 การใช้ฟังก์ชัน array_values ( ) สำหรับอ่านค่าของสมาชิกในอาร์เรย์ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $array เป็นตัวแปรชนิดอาร์เรย์ มีค่าตามลำดับ ดังนี้

$array ["size"] มีค่าเท่ากับ "XL"

$array ["color"] มีค่าเท่ากับ "gold"

บรรทัดที่ 3 แสดงผลลัพธ์เป็นอาร์เรย์ของค่าที่อ่านได้จากสมาชิกอาร์เรย์ทั้งหมด ของตัวแปร $array คือ Array ([0] => XL [1] => gold) หมายความว่า ไม่อ่านค่าคีย์ของอาร์เรย์ อ่านเฉพาะค่าของสมาชิกแต่ละตัวในอาร์เรย์

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

3) การใช้ฟังก์ชัน array_unique ( )

ฟังก์ชัน array_unique ( ) ใช้สำหรับอ่านค่าของสมาชิกในอาร์เรย์ โดยเลือกเฉพาะข้อมูลที่ไม่ซ้ำกัน (ตัวพิมพ์เล็กและตัวพิมพ์ใหญ่สำหรับภาษาอังกฤษถือว่าไม่เหมือนกัน)

รูปแบบ

```
array array_unique ( array $array [, int $sort_flags = SORT_STRING ] )
```

เมื่อ $array หมายถึง ตัวแปรอาร์เรย์ที่ต้องการอ่านค่า

$sort_flags หมายถึง พารามิเตอร์ที่ใช้ร่วมกับการจัดเรียงตามชนิด โดยจะต้องกำหนดร่วมกับค่า SORT_STRING ของการจัดเรียง (จะกำหนดหรือไม่ก็ได้) ดังนี้

SORT_REGULAR หมายถึง จัดเรียงโดยการเปรียบเทียบตามปกติ (ไม่มีการเปลี่ยนชนิด)

SORT_NUMERIC หมายถึง จัดเรียงโดยการเปรียบเทียบระหว่างค่าที่เป็นชนิดตัวเลข

SORT_STRING หมายถึง จัดเรียงโดยการเปรียบเทียบระหว่างค่าที่เป็นชนิดตัวอักษรหรือข้อความ

ตัวอย่างที่ 6.13 การใช้ฟังก์ชัน array_unique ( ) สำหรับอ่านค่าของสมาชิกในอาร์เรย์ที่ไม่ซ้ำกัน

```
1  <?php
2      $input = array ("a" => "green", "red", "b" => "green", "blue", "red");
3      $result = array_unique ($input);
4      print_r ($result);
5  ?>
```

ผลลัพธ์

```
Array ( [a] => green [0] => red [1] => blue )
```

6.3.2 ฟังก์ชันในการเพิ่มสมาชิกในอาร์เรย์

1) การใช้ฟังก์ชัน array_push ( )

ฟังก์ชัน array_push ( ) ใช้สำหรับเพิ่มสมาชิกลงในอาร์เรย์ โดยข้อมูลที่จะเพิ่มลงไปจะมีจำนวนเท่าไหร่ก็ได้ หรือจะนำมาจากอาร์เรย์อื่นก็ได้

รูปแบบ

```
int array_push ( array &$array , mixed $var [, mixed $... ] )
```

เมื่อ $array หมายถึง ตัวแปรอาร์เรย์ที่จะนำข้อมูลเพิ่ม

$var หมายถึง ข้อมูลที่ต้องการเพิ่ม

ตัวอย่างที่ 6.14 ฟังก์ชัน array_push ( ) สำหรับเพิ่มสมาชิกลงในอาร์เรย์

```
1    <?php
2        $stack = array ("orange", "banana");
3        array_push ($stack, "apple", "raspberry");
4        print_r ($stack);
5    ?>
```

ผลลัพธ์

```
Array ( [0] => orange [1] => banana [2] => apple [3] => raspberry )
```

2) การใช้ฟังก์ชัน array_pad ( )

ฟังก์ชัน array_pad ( ) ใช้สำหรับขยายสมาชิกเพิ่มให้กับอาร์เรย์

รูปแบบ

```
array array_pad ( array $input , int $pad_size , mixed $pad_value )
```

เมื่อ $input หมายถึง ตัวแปรอาร์เรย์ที่จะขยายเพิ่มสมาชิก

$pad_size หมายถึง ขนาดใหม่ของตัวแปรอาร์เรย์

$pad_value หมายถึง ข้อมูลที่จะเพิ่ม

ตัวอย่างที่ 6.15 ฟังก์ชัน array_pad ( ) สำหรับขยายสมาชิกเพิ่มให้กับอาร์เรย์

```
1    <?php
2        $input = array (12, 10, 9);
3        $result = array_pad ($input, 5, 0);
4        $result = array_pad ($input, -7, -1);
5        $result = array_pad ($input, 2, "noop");
6    ?>
```

จากตัวอย่างที่ 6.15 ฟังก์ชัน array_pad ( ) สำหรับขยายสมาชิกเพิ่มให้กับอาร์เรย์ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $input เป็นตัวแปรชนิดอาร์เรย์ มีค่า 12, 10 และ 9 ตามลำดับ

บรรทัดที่ 3 กำหนดให้ตัวแปร $result เป็นตัวแปรชนิดอาร์เรย์ มีค่าเท่ากับ array (12, 10, 9, 0, 0) เพราะผลของการเรียกใช้ฟังก์ชัน array_pad ($input, 5, 0) โดยพารามิเตอร์ 5 หมายความว่า ขยายขนาดใหม่ของตัวแปรอาร์เรย์ทางด้านขวา (ขยายเพิ่มหลังจากค่าในตัวแปร $input) และ 0 หมายความว่า ข้อมูลที่จะกำหนดให้สมาชิกใหม่มีค่าเท่ากับ 0

บรรทัดที่ 4 กำหนดให้ตัวแปร $result เป็นตัวแปรชนิดอาร์เรย์ มีค่าเท่ากับ array (-1, -1, -1, -1, 12, 10, 9) เพราะผลของการเรียกใช้ฟังก์ชัน array_pad ($input, -7, -1) โดยพารามิเตอร์ -7 หมายความว่า ขยายขนาดใหม่ของตัวแปรอาร์เรย์ทางด้านซ้าย (ขยายเพิ่มก่อนค่าตัวแปร $input) และ -1 หมายความว่า ข้อมูลที่จะกำหนดให้สมาชิกใหม่มีค่าเท่ากับ -1

บรรทัดที่ 5 ไม่สามารถกำหนดค่าให้กับตัวแปร $result ได้ เนื่องจากไม่สามารถขยาย (not padded) เพราะกำหนดพารามิเตอร์ $pad_size มีขนาดน้อยกว่า $input

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

3) การใช้ฟังก์ชัน array_unshift ( )

ฟังก์ชัน array_unshift ( ) ใช้สำหรับเพิ่มสมาชิกในตำแหน่งแรกของตัวแปรอาร์เรย์

รูปแบบ

```
int array_unshift ( array &$array , mixed $var [, mixed $... ] )
```

เมื่อ $array หมายถึง ตัวแปรอาร์เรย์ที่จะเพิ่มสมาชิก

$var หมายถึง ข้อมูลหรือค่าที่จะเพิ่ม

ตัวอย่างที่ 6.16 ฟังก์ชัน array_unshift ( ) สำหรับเพิ่มสมาชิกตำแหน่งแรกของอาร์เรย์

```
1   <?php
2       $queue = array ("orange", "banana");
3       array_unshift ($queue, "apple", "raspberry");
4       print_r ($queue);
5   ?>
```

ผลลัพธ์

```
Array ( [0] => apple [1] => raspberry [2] => orange [3] => banana )
```

4) การใช้ฟังก์ชัน array_slice ( )

ฟังก์ชัน array_slice ( ) ใช้สำหรับลบ แทนที่ และเพิ่ม สมาชิกในตำแหน่งใดๆ ของอาร์เรย์

รูปแบบ

```
array array_splice ( array &$input , int $offset [, int $length = 0 [, mixed $replacement ]] )
```

เมื่อ $input หมายถึง ตัวแปรอาร์เรย์ที่จะนำข้อมูลเพิ่ม

$length หมายถึง จำนวนสมาชิกที่จะลบ (จะกำหนดหรือไม่ก็ได้) แบ่งเป็น 3 กรณี คือ 1) หากกำหนด จะลบสมาชิกตามจำนวนที่กำหนด โดยเริ่มจากตำแหน่ง offset ที่ระบุ 2) หากไม่กำหนด จะลบสมาชิกตั้งแต่ตำแหน่ง offset ที่ระบุจนถึงสมาชิกสุดท้ายของอาร์เรย์

3) หากกำหนดเป็น 0 คือ ไม่ต้องการลบสมาชิกออก แต่ต้องการเพิ่มสมาชิกลงในตำแหน่ง offset ที่ระบุ แต่ต้องกำหนด replacement ด้วย

$replacement หมายถึง ข้อมูลที่จะเพิ่มลงไปยังตำแหน่ง offset ที่ระบุ

ตัวอย่างที่ 6.17 ฟังก์ชัน array_slice ( ) สำหรับลบ แทนที่ และเพิ่ม สมาชิกในตำแหน่งใดๆ ของอาร์เรย์

```
1   <?php
2       $input = array ("red", "green", "blue", "yellow");
3       array_splice ($input, 2);
4       print_r ($input);
5   ?>
```

ผลลัพธ์

```
Array ( [0] => red [1] => green )
```

6.3.3 ฟังก์ชันในการลบสมาชิกในอาร์เรย์

1) การใช้ฟังก์ชัน unset ( )

ฟังก์ชัน unset ( ) ใช้สำหรับลบหรือยกเลิกตัวแปรทั่วไปหรือตัวแปรชนิดอาร์เรย์

รูปแบบ

```
void unset ( mixed $var [, mixed $... ] )
```

เมื่อ $var หมายถึง ตัวแปรอาร์เรย์ที่ต้องการลบ

ตัวอย่างที่ 6.18 ฟังก์ชัน unset ( ) สำหรับลบหรือยกเลิกตัวแปรทั่วไปหรือตัวแปรชนิดอาร์เรย์

```
1   <?php
2       $stack = array ("orange", "banana");
3       unset ($stack [1]);
4       print_r ($stack);
5   ?>
```

ผลลัพธ์

```
Array ( [0] => orange )
```

2) การใช้ฟังก์ชัน array_pop ( )

ฟังก์ชัน array_pop ( ) ใช้สำหรับลบสมาชิกตำแหน่งสุดท้ายของตัวแปรชนิดอาร์เรย์ หากตัวแปรเป็นค่าว่างหรือตัวแปรที่ส่งเข้าไปไม่ใช่ตัวแปรชนิดอาร์เรย์ ฟังก์ชันจะส่งค่ากลับเป็น Null หากสามารถลบได้จะส่งค่ากลับเป็นค่าลำดับสุดท้ายของสมาชิกอาร์เรย์

รูปแบบ

```
mixed array_pop ( array &$array )
```

เมื่อ $array หมายถึง ตัวแปรอาร์เรย์ที่ต้องการลบ

ตัวอย่างที่ 6.19 ฟังก์ชัน array_pop ( ) สำหรับลบสมาชิกตำแหน่งสุดท้ายของตัวแปรอาร์เรย์

```
1   <?php
2       $stack= array ("orange", "banana", "apple");
3       array_pop ($stack);
4       print_r ($stack);
5   ?>
```

ผลลัพธ์

```
Array ( [0] => orange [1] => banana )
```

3) การใช้ฟังก์ชัน array_shift ( )

ฟังก์ชัน array_shift ( ) ใช้สำหรับลบสมาชิกตำแหน่งแรกของตัวแปรชนิดอาร์เรย์ และย้ายลำดับของสมาชิกใหม่โดยเริ่มต้นจาก 0

รูปแบบ

```
mixed array_shift ( array &$array )
```

เมื่อ $array หมายถึง ตัวแปรอาร์เรย์ที่ต้องการลบ

ตัวอย่างที่ 6.20 ฟังก์ชัน array_shift ( ) สำหรับลบสมาชิกตำแหน่งแรกของตัวแปรชนิดอาร์เรย์

```
1   <?php
2       $stack = array ("orange", "banana", "apple");
3       array_shift ($stack);
3       print_r ($stack);
4   ?>
```

ผลลัพธ์

```
Array ( [0] => banana [1] => apple )
```

6.3.4 ฟังก์ชันในการนับจำนวนสมาชิกในอาร์เรย์

1) การใช้ฟังก์ชัน count ( )

ฟังก์ชัน count ( ) ใช้สำหรับนับจำนวนสมาชิกทั้งหมดในอาร์เรย์

รูปแบบ

```
int count ( mixed $var [, int $mode = COUNT_NORMAL ] )
```

เมื่อ $var หมายถึง ตัวแปรอาร์เรย์ที่ต้องการนับจำนวนสมาชิก

$mode หมายถึง พารามิเตอร์เสริมสำหรับนับอาร์เรย์แบบ COUNT_RECURSIVE

หมายเหตุ

ฟังก์ชัน count ( ) จะมีการส่งค่ากลับเป็นเลขจำนวนสมาชิกทั้งหมดในอาร์เรย์ ในกรณีที่ตัวแปรที่ส่งเข้ามาไม่ใช่ตัวแปรชนิดอาร์เรย์ฟังก์ชัน count ( ) จะส่งค่ากลับเป็น 1 และในกรณีที่ตัวแปรอาร์เรย์เป็นค่าว่างจะส่งค่ากลับเป็น 0

ตัวอย่างที่ 6.21 การใช้ฟังก์ชัน count ( ) สำหรับนับจำนวนสมาชิกทั้งหมดในอาร์เรย์

```
1    <?php
2        $a = array ("1","3","6");
3        printf("%d<br>", count($a));
4        printf("%d<br>", count(null));
5        printf("%d<br>", count(false));
6    ?>
```

จากตัวอย่างที่ 6.21 การใช้ฟังก์ชัน count ( ) สำหรับนับจำนวนสมาชิกทั้งหมดในอาร์เรย์ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $a เป็นข้อมูลชนิดอาร์เรย์ ประกอบด้วย 1, 3 และ 6

บรรทัดที่ 3 แสดงค่าผลการนับจำนวนสมาชิกอาร์เรย์ของตัวแปร $a ผลลัพธ์ที่ได้ คือ 3

บรรทัดที่ 4 แสดงค่าผลการนับจำนวนสมาชิกอาร์เรย์ ที่มีค่า null ผลลัพธ์ที่ได้ คือ 0

บรรทัดที่ 5 แสดงค่าผลการนับจำนวนสมาชิกอาร์เรย์ ที่มีค่า false ผลลัพธ์ที่ได้ คือ 1

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

2) การใช้ฟังก์ชัน sizeof ( )

ฟังก์ชัน sizeof ( ) ใช้สำหรับนับจำนวนสมาชิกทั้งหมดในอาร์เรย์ เช่นเดียวกับฟังก์ชัน count ( )

รูปแบบ

```
int sizeof ( mixed $var )
```

เมื่อ $var หมายถึง ตัวแปรอาร์เรย์ที่ต้องการนับจำนวนสมาชิก

ตัวอย่างที่ 6.22 การใช้ฟังก์ชัน sizeof ( ) สำหรับนับจำนวนสมาชิกทั้งหมดในอาร์เรย์

```
1    <?php
2        $a = array ("1","3","6");
3        printf("%d<br>", sizeof ($a));
4    ?>
```

ผลลัพธ์

```
4
```

3) การใช้ฟังก์ชัน array_count_values ( )

ฟังก์ชัน array_count_values ( ) ใช้สำหรับนับจำนวนสมาชิกทั้งหมดในอาร์เรย์ แต่ผลลัพธ์ที่ได้จะเป็นอาร์เรย์ที่แสดงว่า สมาชิกแต่ละตัวมีอยู่จำนวนเท่าไหร่ โดยค่าคีย์จะเป็นสมาชิกแต่ละตัว และค่าจะเป็นจำนวนที่นับได้

รูปแบบ

```
array array_count_values ( array $input )
```

เมื่อ $input หมายถึง ตัวแปรอาร์เรย์ที่ต้องการนับ

ตัวอย่างที่ 6.23 การใช้ฟังก์ชัน array_count_values ( ) สำหรับนับจำนวนสมาชิกทั้งหมดในอาร์เรย์

```
1   <?php
2       $array = array (1, "hello", 1, "world", "hello");
3       print_r (array_count_values ($array));
4   ?>
```

ผลลัพธ์

```
Array ( [1] => 2 [hello] => 2 [world] => 1 )
```

6.3.5 ฟังก์ชันในการเรียงลำดับข้อมูลในอาร์เรย์โดยใช้อินเด็กซ์

ฟังก์ชันในการเรียงลำดับข้อมูลในอาร์เรย์โดยใช้อินเด็กซ์ จะมี 2 ฟังก์ชันให้เลือกใช้งาน คือ ฟังก์ชัน sort ( ) และ rsort ( ) มีรายละเอียด ดังนี้

1) การใช้ฟังก์ชัน sort ( )

ฟังก์ชัน sort ( ) ใช้สำหรับเรียงลำดับข้อมูลในอาร์เรย์โดยใช้อินเด็กซ์ เรียงลำดับข้อมูลจากน้อยไม่หามาก มีรูปแบบ ดังนี้

รูปแบบ

```
bool sort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

เมื่อ $array หมายถึง ตัวแปรอาร์เรย์ที่ต้องการจัดเรียงและมีอินเด็กซ์เป็นตัวเลข

$sort_flags หมายถึง พารามิเตอร์เสริมสำหรับการจัดเรียงเพิ่มเติม โดยการระบุสตริงสำหรับการจัดเรียง มีรายละเอียด (จะกำหนดหรือไม่ก็ได้) ดังนี้

- SORT_REGULAR จัดเรียงโดยการเปรียบเทียบระหว่างค่าตามปกติ (ไม่ต้องเปลี่ยนค่าคีย์)
- SORT_NUMERIC จัดเรียงโดยการเปรียบเทียบระหว่างค่าที่เป็นชนิดตัวเลข

- SORT_STRING จัดเรียงโดยการเปรียบเทียบระหว่างค่าที่เป็นตัวอักษรหรือข้อความ
- SORT_NATURAL จัดเรียงโดยการเปรียบเทียบระหว่างค่าที่เป็นตัวอักษรหรือข้อความเหมือนกับฟังก์ชัน natsort ( ) ซึ่งใช้สำหรับการจัดเรียงข้อมูลในอาร์เรย์ที่มีค่าอินเด็กซ์เป็นชนิดตัวอักษรหรือข้อความ
- SORT_FLAG_CASE รูปแบบการจัดเรียงที่สามารถใช้ร่วมกับการจัดเรียงแบบ SORT_STRING หรือ SORT_NATURAL โดยการเปรียบเทียบในลักษณะตัวอักษรพิมพ์เล็กพิมพ์ใหญ่ถือว่าเป็นตัวเดียวกัน (case-insensitively)

ตัวอย่างที่ 6.24 การใช้ฟังก์ชัน sort ( )

```
1   <?php
2       $fruits = array ("lemon", "banana", "apple");
3       sort ($fruits);
4       foreach ($fruits as $key => $val) echo "fruits[" . $key . "] = " . $val . "<br>";
5   ?>
```

ผลลัพธ์

```
fruits [0] = apple
fruits [1] = banana
fruits [2] = lemon
```

ตัวอย่างที่ 6.25 การใช้ฟังก์ชัน sort ( ) โดยกำหนดการจัดเรียงแบบ case-insensitive natural ordering

```
1   <?php
2       $fruits = array ( "Orange1", "orange2", "Orange3", "orange20" );
3       @sort($fruits, SORT_NATURAL | SORT_FLAG_CASE);
4       foreach ($fruits as $key => $val)  echo "fruits[" . $key . "] = " . $val . "<br>";
5   ?>
```

ผลลัพธ์

```
fruits [0] = Orange1
fruits [1] = orange2
fruits [2] = Orange3
fruits [3] = orange20
```

2) การใช้ฟังก์ชัน rsort ( )

ฟังก์ชัน rsort ( ) ใช้สำหรับเรียงลำดับข้อมูลในอาร์เรย์โดยใช้อินเด็กซ์ เรียงลำดับข้อมูลจากมากไปหาน้อย มีรูปแบบเหมือนกับฟังก์ชัน sort ( ) มีตัวอย่างดังนี้

ตัวอย่างที่ 6.26 การใช้ฟังก์ชัน rsort ( )

```
1    <?php
2        $fruits = array ("lemon", "orange", "banana", "apple");
3        rsort ($fruits);
4        foreach ($fruits as $key => $val) echo "$key = $val";
5    ?>
```

ผลลัพธ์

```
0 = orange 1 = lemon 2 = banana 3 = apple
```

6.3.6 ฟังก์ชันที่ใช้ในการเรียงลำดับข้อมูลในอาร์เรย์โดยใช้คีย์อาร์เรย์

ฟังก์ชันในการเรียงลำดับข้อมูลในอาร์เรย์โดยใช้คีย์อาร์เรย์ มี 4 ฟังก์ชันให้เลือกใช้งาน ประกอบด้วย 1) asort ( ) 2) arsort ( ) 3) ksort ( ) และ 4) krsort ( ) มีรายละเอียด ดังนี้

1) การใช้ฟังก์ชัน asort ( )

ฟังก์ชัน asort ( ) เป็นฟังก์ชันที่ใช้สำหรับจัดเรียงลำดับข้อมูลในอาร์เรย์โดยพิจารณาจากค่าหรือข้อมูลในตัวแปรอาร์เรย์ จากน้อยไปหามาก มีรูปแบบการใช้งาน ดังนี้

รูปแบบ

```
bool asort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

เมื่อ $array หมายถึง ตัวแปรอาร์เรย์ที่ต้องการจัดเรียง

$sort_flags หมายถึง พารามิเตอร์เสริมสำหรับการจัดเรียงเพิ่มเติม โดยการระบุสตริงสำหรับการจัดเรียง จะมีรูปแบบเหมือนฟังก์ชันจัดเรียงก่อนหน้า

ตัวอย่างที่ 6.27 การใช้ฟังก์ชัน asort ( ) สำหรับจัดเรียงลำดับข้อมูลในอาร์เรย์

```
1    <?php
2        $fruits = array("d" => "lemon", "b" => "banana", "c" => "apple");
3        asort ($fruits);
4        foreach ($fruits as $key => $val)  echo "$key = $val <br>";
5    ?>
```

ผลลัพธ์

```
c = apple
b = banana
d = lemon
```

2) การใช้ฟังก์ชัน arsort ( )

ฟังก์ชัน arsort ( ) เป็นฟังก์ชันที่ใช้สำหรับจัดเรียงลำดับข้อมูลในอาร์เรย์โดยพิจารณาจากค่าหรือข้อมูลในตัวแปรอาร์เรย์ จากมากไปหาน้อย มีรูปแบบการใช้งานเหมือนกับฟังก์ชัน asort ( )

ตัวอย่างที่ 6.28 การใช้ฟังก์ชัน arsort ( ) เป็นฟังก์ชันที่ใช้สำหรับจัดเรียงลำดับข้อมูลในอาร์เรย์

```
1   <?php
2       $fruits = array("d" => "lemon", "b" => "banana", "c" => "apple");
3       arsort ($fruits);
4       foreach ($fruits as $key => $val)  echo "$key = $val <br>";
5   ?>
```

ผลลัพธ์

```
d = lemon
b = banana
c = apple
```

3) การใช้ฟังก์ชัน ksort ( )

ฟังก์ชัน ksort ( ) เป็นฟังก์ชันที่ใช้สำหรับจัดเรียงลำดับข้อมูลในอาร์เรย์โดยพิจารณาจากคีย์ของตัวแปรอาร์เรย์ จากน้อยไปหามาก มีรูปแบบการใช้งาน ดังนี้

รูปแบบ

```
bool ksort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

เมื่อ $array หมายถึง ตัวแปรอาร์เรย์ที่ต้องการจัดเรียง

$sort_flags หมายถึง พารามิเตอร์เสริมสำหรับการจัดเรียงเพิ่มเติม โดยการระบุสตริงสำหรับการจัดเรียง จะมีรูปแบบเหมือนฟังก์ชันจัดเรียงก่อนหน้า

ตัวอย่างที่ 6.29 การใช้ฟังก์ชัน ksort ( ) สำหรับจัดเรียงลำดับข้อมูลในอาร์เรย์

```
1   <?php
2       $fruits = array ("d" => "lemon", "b" => "banana", "c" => "apple");
3       ksort ($fruits);
4       foreach ($fruits as $key => $val) echo "$key = $val <br>";
5   ?>
```

ผลลัพธ์

```
b = banana
c = apple
d = lemon
```

4) การใช้ฟังก์ชัน krsort ( )

ฟังก์ชัน krsort ( ) เป็นฟังก์ชันที่ใช้สำหรับจัดเรียงลำดับข้อมูลในอาร์เรย์โดยพิจารณาจากคีย์ของตัวแปรอาร์เรย์ จากมากไปหาน้อย มีรูปแบบเหมือนกับฟังก์ชัน ksort ( ) มีตัวอย่างดังนี้

ตัวอย่างที่ 6.30 การใช้ฟังก์ชัน krsort ( ) สำหรับจัดเรียงลำดับข้อมูลในอาร์เรย์

```
1   <?php
2       $fruits = array ("d" => "lemon", "b" => "banana", "c" => "apple");
3       krsort ($fruits);
4       foreach ($fruits as $key => $val) echo "$key = $val <br>";
5   ?>
```

ผลลัพธ์

```
d = lemon
c = apple
b = banana
```

โดยสรุป ทั้ง 4 ฟังก์ชัน เป็นฟังก์ชันที่ใช้สำหรับเรียงลำดับข้อมูลในอาร์เรย์โดยใช้คีย์อาร์เรย์ สามารถจำแนกได้เป็นกลุ่มใหญ่ ดังนี้

ตารางที่ 6.2 แสดงผลสรุปฟังก์ชันที่ใช้สำหรับเรียงลำดับข้อมูลในอาร์เรย์โดยใช้คีย์อาร์เรย์

เรียงจาก	ฟังก์ชัน	ลำดับการเรียง
ข้อมูล (Value)	asort ( )	น้อยไปหามาก
	arsort ( )	มากไปหาน้อย
อินเด็กซ์หรือคีย์ (Index or Key)	ksort ( )	น้อยไปหามาก
	krsort ( )	มากไปหาน้อย

6.3.7 ฟังก์ชันที่ใช้สำหรับการจัดการตัวชี้ตำแหน่งสมาชิกในอาร์เรย์

เมื่อเพิ่มหรือกำหนดค่าข้อมูลในอาร์เรย์ ตำแหน่งของตัวชี้ตำแหน่งสมาชิกในอาร์เรย์จะชี้อยู่ที่ตำแหน่งของสมาชิกปัจจุบัน หากต้องการกำหนดให้ตัวชี้ตำแหน่งสมาชิกในอาร์เรย์ชี้ไปที่ตำแหน่งต่างๆ ของอาร์เรย์ สามารถทำได้โดยใช้ฟังก์ชันต่างๆ ดังตารางที่ 6.3

ตารางที่ 6.3 สรุปฟังก์ชันที่ใช้สำหรับการจัดการตัวชี้ตำแหน่งสมาชิกในอาร์เรย์

ฟังก์ชัน	การทำงาน
each ( )	ฟังก์ชันสำหรับเลื่อนตัวชี้ตำแหน่งสมาชิกในอาร์เรย์ไปข้างหน้าครั้งละ 1 ตำแหน่ง หากเรียกใช้ฟังก์ชัน each ( ) จะคืนค่าสมาชิกก่อนหน้าที่จะเลื่อนตัวชี้ตำแหน่งสมาชิกในอาร์เรย์
current ( )	ฟังก์ชันที่ใช้สำหรับคืนค่าสมาชิกปัจจุบันที่ตัวชี้ตำแหน่งสมาชิกในอาร์เรย์กำลังชี้อยู่ หากก่อนหน้ายังไม่มีการเรียกใช้ฟังก์ชันสำหรับการเลื่อนตัวชี้ตำแหน่งสมาชิกในอาร์เรย์หลังจากอาร์เรย์ถูกสร้าง ตำแหน่งของตัวชี้ตำแหน่งสมาชิกในอาร์เรย์ปัจจุบัน คือ ตำแหน่งแรกของสมาชิกอาร์เรย์
reset ( )	ฟังก์ชันที่ใช้สำหรับเลื่อนตัวชี้ตำแหน่งสมาชิกในอาร์เรย์ไปยังสมาชิกตำแหน่งแรกของอาร์เรย์
end ( )	ฟังก์ชันที่ใช้สำหรับเลื่อนตัวชี้ตำแหน่งสมาชิกในอาร์เรย์ไปยังสมาชิกตำแหน่งสุดท้ายของอาร์เรย์
next ( )	ฟังก์ชันสำหรับเลื่อนตัวชี้ตำแหน่งสมาชิกในอาร์เรย์ไปข้างหน้าครั้งละ 1 ตำแหน่ง หากเรียกใช้ฟังก์ชัน next ( ) จะเลื่อนตัวชี้ตำแหน่งสมาชิกในอาร์เรย์ก่อนแล้วจึงคืนค่าสมาชิกก่อนหน้า
prev ( )	ฟังก์ชันสำหรับเลื่อนตัวชี้ตำแหน่งสมาชิกในอาร์เรย์ตรงข้ามกับฟังก์ชัน next ( ) คือ ถอยกลับครั้งละ 1 ตำแหน่ง หากเรียกใช้ฟังก์ชัน prev ( ) จะเลื่อนตัวชี้ตำแหน่งสมาชิกในอาร์เรย์ก่อนแล้วจึงคืนค่าสมาชิกก่อนหน้า

6.3.8 ฟังก์ชันสำหรับการรวมอาร์เรย์

ฟังก์ชันสำหรับการรวมอาร์เรย์ มีให้เลือกใช้งาน คือ 1) ฟังก์ชัน array_merge ( ) 2) array_merge_recursive ( ) และ 3) array_combine ( )

ทั้ง 2 ฟังก์ชันนี้ใช้สำหรับรวมอาร์เรย์เหมือนกัน หลักการรวมอาร์เรย์ คือ จะนำค่าข้อมูลมาเก็บในลักษณะของสมาชิกต่อกันไปเรื่อยๆ ในรูปแบบของอาร์เรย์หลายมิติ แล้วสร้างอินเด็กซ์ชนิดตัวเลขให้ใหม่เรียงตามลำดับ ข้อแตกต่างระหว่าง 2 ฟังก์ชันนี้ คือ

1) ฟังก์ชัน array_merge ( ) หากอาร์เรย์ที่นำมารวมมีอินเด็กซ์หรือคีย์ที่เป็นสตริงซ้ำกัน จะนำค่าข้อมูลของอาร์เรย์ชุดหลังทับค่าข้อมูลของอาร์เรย์ชุดแรก

2) ฟังก์ชัน array_merge_recursive ( ) จะนำข้อมูลของอาร์เรย์ชุดหลังมาต่อท้ายกับค่าข้อมูลตัวแรก ไม่ว่าค่าข้อมูลหรืออินเด็กซ์หรือคีย์จะซ้ำกันหรือไม่

รูปแบบ ฟังก์ชัน array_merge ( )

```
array array_merge ( array $array1 [, array $... ] )
```

รูปแบบ ฟังก์ชัน array_merge_recursive ( )

```
array array_merge_recursive ( array $array1 [, array $... ] )
```

เมื่อ $array1 หมายถึง ตัวแปรอาร์เรย์ที่ต้องการนำมารวมลำดับที่ 1

$... หมายถึง ตัวแปรอาร์เรย์ที่ต้องการนำมารวมลำดับต่อๆ ไป

ตัวอย่างที่ 6.31 การใช้ฟังก์ชัน array_merge ( ) สำหรับรวมอาร์เรย์

```
1   <?php
2       $array1 = array ("color" => "red", 2, 4);
3       $array2 = array ("a", "b", "color" => "green", "shape" => "trapezoid", 4);
4       $result = array_merge ($array1, $array2);
5       print_r ($result);
6   ?>
```

ผลลัพธ์

หมายเหตุ หากค่าอินเด็กซ์หรือคีย์ซ้ำกันข้อมูลของอาร์เรย์ตัวแรกจะถูกทับด้วยข้อมูลอาร์เรย์ล่าสุด

```
Array ( [color] => green [0] => 2 [1] => 4 [2] => a [3] => b [shape] => trapezoid [4] => 4 )
```

ตัวอย่างที่ 6.32 การใช้ฟังก์ชัน array_merge_recursive ( )

```
1   <?php
2       $array1 = array ("color" => "red", 2, 4);
3       $array2 = array ("a", "b", "color" => "green", "shape" => "trapezoid", 4);
4       $result = array_merge_recursive ($array1, $array2);
5       print_r ($result);
6   ?>
```

ผลลัพธ์

หมายเหตุ หากอินเด็กซ์หรือคีย์ซ้ำกันจะสร้างเป็นอาร์เรย์ซ้อนอาร์เรย์ หรือเรียกว่าอาร์เรย์หลายมิติ

```
Array ( [color] => Array ( [0] => red [1] => green ) [0] => 2 [1] => 4 [2] => a [3] => b
[shape] => trapezoid [4] => 4 )
```

3) ฟังก์ชัน array_combine ( )

ฟังก์ชัน array_combine ( ) เป็นฟังก์ชันที่ใช้สำหรับรวมอาร์เรย์ 2 อาร์เรย์เข้าด้วยกัน โดยใช้ค่าข้อมูลของอาร์เรย์ตัวแรกเป็นอินเด็กซ์หรือคีย์ ส่วนค่าข้อมูลของอาร์เรย์ตัวที่สองเป็นค่าข้อมูลของอาร์เรย์ใหม่ที่เกิดจากการรวมกันของอาร์เรย์ทั้งสอง ถ้าจำนวนของสมาชิกของทั้งสองอาร์เรย์ไม่เท่ากัน หรืออาร์เรย์ตัวใดตัวหนึ่งเป็นค่าว่าง ฟังก์ชันจะส่งคืนค่าเป็นเท็จ (False)

รูปแบบ

```
array array_combine ( array $keys , array $values )
```

เมื่อ $key คือ ตัวแปรอาร์เรย์ที่ใช้สำหรับเป็นอินเด็กซ์หรือคีย์

$values คือ ตัวแปรอาร์เรย์ที่ใช้สำหรับกำหนดค่า

ตัวอย่างที่ 6.33 การใช้ฟังก์ชัน array_combine ( ) สำหรับรวมอาร์เรย์ 2 อาร์เรย์เข้าด้วยกัน

```
1   <?php
2       $a = array ('green', 'red', 'yellow');
3       $b = array ('avocado', 'apple', 'banana');
4       $c = array_combine ($a, $b);
5       print_r ($c);
6   ?>
```

ผลลัพธ์

```
Array ( [green] => avocado [red] => apple [yellow] => banana )
```

6.3.9 ฟังก์ชันสำหรับการสลับค่าระหว่างอินเด็กซ์หรือคีย์กับค่าข้อมูลในอาร์เรย์

ฟังก์ชันสำหรับการสลับค่าระหว่างอินเด็กซ์หรือคีย์กับค่าข้อมูลในอาร์เรย์ คือ ฟังก์ชัน array_flip ( ) มีรูปแบบดังนี้

รูปแบบ

```
array array_flip ( array $trans )
```

เมื่อ $trans หมายถึง ตัวแปรอาร์เรย์ที่ต้องการสลับค่า

ตัวอย่างที่ 6.34 การใช้ฟังก์ชัน array_flip ( ) สำหรับสลับค่าระหว่างอินเด็กซ์หรือคีย์กับค่าข้อมูล

```
1   <?php
2       $trans = array ("a" => 1, "b" => 1, "c" => 2);
3       $trans = array_flip ($trans);
4       print_r ($trans);
5   ?>
```

ผลลัพธ์

```
Array ( [1] => b [2] => c )
```

6.3.10 ฟังก์ชันสำหรับการค้นหาอินเด็กซ์หรือคีย์และค่าข้อมูลในอาร์เรย์

1) การใช้ฟังก์ชัน array_key_exists ( )

ฟังก์ชัน array_key_exists ( ) เป็นฟังก์ชันที่ใช้สำหรับหาอินเด็กซ์หรือคีย์ที่ระบุมีอยู่ในอาร์เรย์หรือไม่ หากพบอินเด็กซ์หรือคีย์ที่ค้นหา ฟังก์ชันจะคืนค่าเป็นจริง (True) หากไม่พบจะคืนค่าเป็นเท็จ (False)

รูปแบบ

```
bool array_key_exists ( mixed $key , array $search )
```

เมื่อ $key หมายถึง อินเด็กซ์หรือคีย์ที่ค้นหา

$search หมายถึง อาร์เรย์ที่นำมาใช้ค้นหา

ตัวอย่างที่ 6.35 การใช้ฟังก์ชัน array_key_exists ( ) สำหรับหาอินเด็กซ์หรือคีย์ที่ระบุมีอยู่ในอาร์เรย์หรือไม่

```
1   <?php
2       $search_array = array ('first' => 1, 'second' => 4);
3       if (array_key_exists ('first', $search_array)) echo "พบข้อมูลในอาร์เรย์";
4       else  echo "ไม่พบข้อมูลในอาร์เรย์";
5   ?>
```

ผลลัพธ์

```
พบข้อมูลในอาร์เรย์
```

2) การใช้ฟังก์ชัน in_array ( ) และ array_search ( )

ฟังก์ชัน in_array ( ) และ array_search ( ) เป็นฟังก์ชันที่ใช้สำหรับหาค่าข้อมูลที่ระบุมีอยู่ในอาร์เรย์หรือไม่ หากฟังก์ชัน in_array ( ) และ array_search ( ) พบค่าข้อมูลที่ค้นหา ฟังก์ชันจะคืนค่าเป็นจริง (True) หากไม่พบจะคืนค่าเป็นเท็จ (False) ซึ่งทั้ง 2 ฟังก์ชันมีรูปแบบเหมือนกัน ดังนี้

รูปแบบ

```
bool in_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] )
```

เมื่อ $needle หมายถึง ค่าข้อมูลที่ต้องการค้นหา

$haystack หมายถึง ตัวแปรอาร์เรย์ที่นำมาใช้ในการค้นหา

$strict หมายถึง ใช้สำหรับการเปรียบเทียบ (===) ใช้ร่วมกับ $needle เพื่อเปรียบเทียบค่าของอาร์เรย์กับค่าที่ค้นหา

ตัวอย่างที่ 6.36 การใช้ฟังก์ชัน in_array ( ) และ array_search ( ) สำหรับหาค่าข้อมูลที่ระบุมีอยู่ในอาร์เรย์หรือไม่

```
1    <?php
2        $os = array ("Mac", "NT", "Irix", "Linux");
3        if (in_array ("Irix", $os)) echo "Got Irix";
4        if (in_array ("mac", $os)) echo "Got mac";
5    ?>
```

ผลลัพธ์

```
Got Irix
```

จากตัวอย่างในเงื่อนไขที่ 2 จะพบว่าเงื่อนไขไม่ถูกต้องเพราะในฟังก์ชัน in_array ( ) จะมีการตรวจสอบอักษรพิมพ์เล็กและพิมพ์ใหญ่ (case-sensitive)

6.3.11 ฟังก์ชันสำหรับการหาค่าข้อมูลสมาชิกที่เหมือนและแตกต่างกันในอาร์เรย์

1) การใช้ฟังก์ชัน array_intersect ( )

ฟังก์ชัน array_intersect ( ) เป็นฟังก์ชันสำหรับหาค่าข้อมูลสมาชิกในอาร์เรย์ ตั้งแต่ 2 อาร์เรย์ขึ้นไป มาทำการ Intersection หาสมาชิกที่มีซ้ำกัน เพื่อกำหนดค่าข้อมูลให้กับอาร์เรย์ใหม่

รูปแบบ

```
array array_intersect ( array $array1 , array $array2 [, array $ ... ] )
```

เมื่อ $array1 หมายถึง ตัวแปรอาร์เรย์ตัวที่ 1

$array2 หมายถึง ตัวแปรอาร์เรย์ตัวที่ 2

$ ... หมายถึง ตัวแปรอาร์เรย์ลำดับต่อๆ ไป

ตัวอย่างที่ 6.37 การใช้ฟังก์ชัน array_intersect ( )

```
1    <?php
2        $array1 = array ("a" => "green", "red", "blue");
3        $array2 = array ("b" => "green", "yellow", "red");
4        $result = array_intersect ($array1, $array2);
5        print_r ($result);
6    ?>
```

ผลลัพธ์

```
Array ( [a] => green [0] => red )
```

2) ฟังก์ชัน array_diff ( )

เป็นฟังก์ชันสำหรับหาค่าข้อมูลสมาชิกในอาร์เรย์ ตั้งแต่ 2 อาร์เรย์ขึ้นไป มาทำการ Differential หาสมาชิกที่มีเฉพาะในอาร์เรย์ตัวแรก แต่ไม่มีในอาร์เรย์อื่นๆ เพื่อกำหนดค่าข้อมูลให้กับอาร์เรย์ใหม่

รูปแบบ

```
array array_diff ( array $array1 , array $array2 [, array $... ] )
```

เมื่อ $array1 หมายถึง ตัวแปรอาร์เรย์ตัวที่ 1

$array2 หมายถึง ตัวแปรอาร์เรย์ตัวที่ 2

$ ... หมายถึง ตัวแปรอาร์เรย์ลำดับต่อๆ ไป

ตัวอย่างที่ 6.38 การใช้ฟังก์ชัน array_diff ( )

```
1   <?php
2       $array1 = array ("a" => "green", "red", "blue", "red");
3       $array2 = array ("b" => "green", "yellow", "red");
4       $result = array_diff ($array1, $array2);
5       print_r ($result);
6   ?>
```

ผลลัพธ์

```
Array ( [1] => blue )
```

6.4 การใช้ข้อมูลจากอาร์เรย์ $_FILE

ตัวแปร $_FILE เป็นตัวแปรชนิดอาร์เรย์เพื่อใช้อ้างถึงข้อมูลเกี่ยวกับการอัพโหลดไฟล์ (File Upload) ระหว่างไคลเอนต์กับเว็บเซิร์ฟเวอร์ มีรายละเอียด ดังตารางที่ 6.4

ตารางที่ 6.4 แสดงค่าคีย์ของอาร์เรย์ $_FILE

คีย์	ค่าข้อมูลและความหมาย
$_FILE ['file'] ['type']	ชนิดของไฟล์ที่อัพโหลด เช่น .JPG, .PNG, .doc เป็นต้น
$_FILE ['file'] ['size']	ขนาดของไฟล์
$_FILE ['file'] ['tmp_name']	ตำแหน่งไดเรกทอรีที่เก็บไฟล์ไว้ชั่วคราว
$_FILE ['file'] ['name']	ชื่อไฟล์ที่อัพโหลด

ตารางที่ 6.4 (ต่อ)

คีย์	ค่าข้อมูลและความหมาย
$_FILE ['file'] ['error']	ข้อมูลที่ผิดพลาดจากการอัพโหลด โดยมีการคืนค่า ดังนี้ 0 คือ แสดงว่าไม่มีข้อผิดพลาด 1 คือ ไฟล์ที่อัพโหลดมีขนาดเกินกว่าค่าที่กำหนดใน php.ini (ปกติ 2 MB) 2 คือ ไฟล์มีขนาดเกินค่าที่กำหนดใน MAX_FILE_SIZE ของฟอร์ม 3 คือ ข้อผิดพลาดในการสื่อสารทำให้อัพโหลดไฟล์ไม่ได้ 4 คือ ไม่มีไฟล์

หลังจากนั้นจะใช้ฟังก์ชัน move_uploaded_file ( ) สำหรับในการเคลื่อนย้ายไฟล์จากไดเรกทอรีชั่วคราวไปยังตำแหน่งใหม่ โดยจะใช้ $_FILE['file'] ['tmp_name'] ในการอ้างถึงไดเร็กทอรีชั่วคราว ส่วนตำแหน่งไดเร็กทอรีปลายทาง ถ้าต้องการใช้ชื่อไฟล์เดิมจะใช้ $_FILE ['file'] ['name'] ถ้าต้องการย้ายไปไดเร็กทอรีอื่นก็สามารถระบุได้ ตัวอย่างที่นิยมใช้งานเพื่อไม่ให้เกิดปัญหาชื่อไฟล์ซ้ำกัน คือ ใช้วันที่และเวลาสำหรับกำหนดชื่อไฟล์ มีตัวอย่าง ดังนี้

ตัวอย่างที่ 6.39 การกำหนดชื่อไฟล์โดยใช้วันที่และเวลา

```
$fileName = mktime (date('H'), date('i'), date('s'),date('m'), date('d'), date('Y')).'.jpg';
```

การกำหนดชื่อไฟล์นั้นใช้สำหรับจัดเก็บลงในฐานข้อมูล หรืออ้างอิงอื่นๆ และใช้ฟังก์ชัน

```
move_uploaded_file ($_FILE ["file"] ["tmp_name"] , "../image/news/".$fileName);
```

หรือใช้ฟังก์ชัน

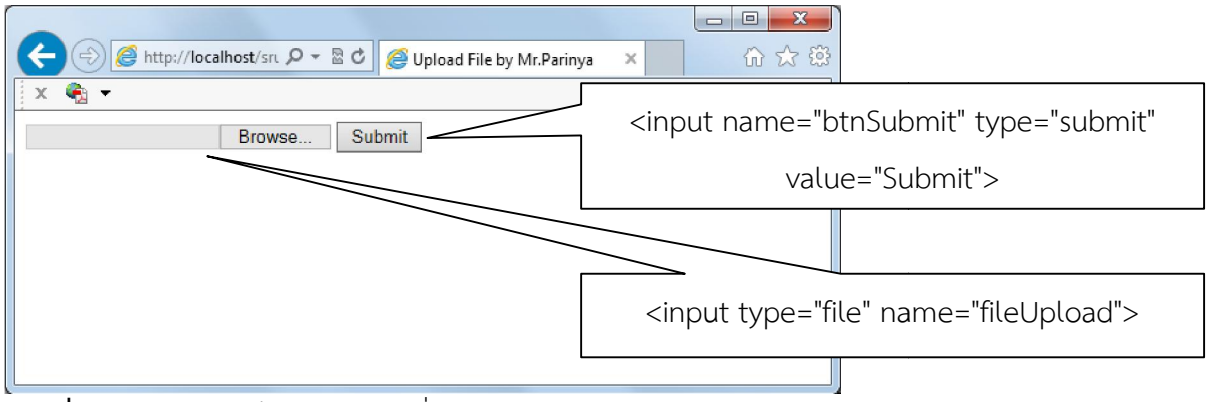
```
copy ($_FILE ["file"] ["tmp_name"] , "../image/news/".$fileName);
```

เพื่อเริ่มต้นอัพโหลดไฟล์

ตัวอย่างที่ 6.40 การประยุกต์พัฒนาเว็บเพจสำหรับอัพโหลดไฟล์

```
<?php /* กำหนดชื่อไฟล์เป็น upload.php*/ ?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Upload File by Mr.Parinya</title>
</head>
<body>
<form name="form1" method="post" action="chk.php" enctype="multipart/form-data">
```

```
        <input type="file" name="fileUpload">
        <input name="btnSubmit" type="submit" value="Submit">
</form>
</body>
</html>
```



ภาพที่ 6.2 แสดงหน้าเว็บเพจสำหรับเริ่มต้นอัพโหลดไฟล์

```
<?php /*กำหนดชื่อไฟล์ chk.php สำหรับเริ่มต้นอัพโหลด ไฟล์ไปเก็บไว้ยังเซิร์ฟเวอร์*/ ?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> Upload File by Mr.Parinya</title>
</head>
<body>
<?php
        if(move_uploaded_file($_FILES["fileUpload"]["tmp_name"],"pictures/".$_FILES["fileU
pload"]["name"])) {      // กำหนดให้ pictures/ คือ โฟลเดอร์ปลายทางสำหรับเก็บไฟล์
            echo "อัพโหลดไฟล์เสร็จเรียบร้อยแล้ว";
        } else {
            echo "ไม่สามารถอัพโหลดไฟล์ได้ กรุณาตรวจสอบใหม่";
        }
 ?>
</body>
</html>
```

หมายเหตุ

ฟังก์ชันอัพโหลดไฟล์ move_uploaded_file ( ) และ copy ( ) มีรูปแบบการใช้งานเหมือนกัน ดังนี้ ชื่อฟังก์ชัน (ตำแหน่งไฟล์ต้นทาง, ตำแหน่งไฟล์ปลายทาง)

เมื่อ ตำแหน่งไฟล์ต้นทาง คือ $_FILE ["file"] ["tmp_name"]

ตำแหน่งไฟล์ปลายทาง คือ ตำแหน่งใหม่ที่จะนำไฟล์ไปเก็บไว้ที่ใดของเครื่องเซิร์ฟเวอร์

ข้อระวังสำหรับการใช้งานทั้งตำแหน่งไฟล์ต้นทางและปลายทาง จำเป็นต้องมีที่ตั้งของไฟล์และชื่อไฟล์เสมอ สำหรับที่ตั้งของไฟล์จะเรียกว่า พาธไดเร็กทอรี (Path Directory)

สรุป

ตัวแปรอาร์เรย์ใช้สำหรับเก็บค่าของข้อมูลที่อยู่ในกลุ่มเดียวกัน หรืออาจเรียกว่าชุดของตัวแปรที่มีการเรียงลำดับที่แน่นอนต่อเนื่องกันไปในหน่วยความจำ ตัวแปรแต่ละตัวจะเรียกว่าสมาชิกของอาร์เรย์ หรืออิลิเมนต์ของอาร์เรย์ สมาชิกแต่ละตัวของอาร์เรย์จะประกอบไปด้วย ค่าข้อมูล และอินเด็กซ์หรือคีย์ สามารถเข้าถึงค่าข้อมูลได้โดยใช้ชื่อตัวแปรเดียวกันตามด้วยอินเด็กซ์หรือคีย์

คำถามท้ายบท

1. จงอธิบายถึงลักษณะของตัวแปรอาร์เรย์ และความแตกต่างระหว่างตัวแปรทั่วไปกับตัวแปรชนิดอาร์เรย์

2. จงอธิบายวิธีและรูปแบบการสร้างอาร์เรย์โดยใช้ฟังก์ชัน array ( ) พร้อมยกตัวอย่างประกอบ

3. จงอธิบายหลักการทำงานของสคริปต์ด้านล่างตั้งแต่บรรทัดที่ 1 ถึงบรรทัดสุดท้าย และบอกผลลัพธ์ที่ได้จากสคริปต์

```
1   <?php
2       $province = array ("สุราษฎร์ธานี", "กระบี่", "ชุมพร", "นครศรีธรรมราช");
3       for ($loop = count ($province)-1; $loop >= 0; $loop--) {
4           printf ("%s <br>", $province [$loop]);
5       }
6   ?>
```

4. จงอธิบายหลักการทำงานของสคริปต์ด้านล่างตั้งแต่บรรทัดที่ 1 ถึงบรรทัดสุดท้าย และบอกผลลัพธ์ที่ได้จากสคริปต์

```
1   <?php
2       $array = array ("number"=>"1","text"=>"test","1", 2, 3);
3       printf("%d<br>", sizeof ($array));
4   ?>
```

5. จงอธิบายหลักการทำงานของสคริปต์ด้านล่างตั้งแต่บรรทัดที่ 1 ถึงบรรทัดสุดท้าย และบอกผลลัพธ์ที่ได้จากสคริปต์

```
1       <?php
2               $array = array (1, "hello", 1, "world", "hello", "number"=> 1, "text"=>"hello");
3               print_r (array_count_values ($array));
4       ?>
```