# บทที่ 2
# PHP เบื้องต้น

เครื่องมือสำหรับสร้างเว็บไซต์เป็นส่วนประกอบสำคัญอย่างหนึ่งในการเขียนสคริปต์ PHP รูปแบบประโยคคำสั่งที่ใช้เขียนจะเรียกว่าสคริปต์คำสั่ง (Statement) ลงในเว็บเพจก่อนอัพโหลดเว็บเพจทั้งหมดไปยังเว็บเซิร์ฟเวอร์ แล้วเรียกดูเว็บเพจที่เครื่องไคลเอ็นท์ด้วยโปรแกรมเว็บบราวเซอร์ เครื่องมือที่ใช้สำหรับเขียนสคริปต์ PHP สามารถเลือกใช้ได้หลายโปรแกรม เช่น Notepad, Notepad++, EditPlus, vi และ Adobe Dreamweaver เป็นต้น รูปแบบการเขียนสคริปต์ PHP นั้น จะใช้เทคนิควิธีการแทรกประโยคคำสั่งร่วมกับ HTML tags และเอกสารดังกล่าวจำเป็นต้องบันทึกเป็นชนิด PHP เท่านั้น ตัวอย่างเช่น index.php เป็นต้น

## 2.1 การแทรกสคริปต์ PHP ในเอกสาร HTML

การแทรกสคริปต์ PHP เพื่อบ่งบอกให้รู้ว่าส่วนที่ระบุนั้นๆ เป็นประโยคคำสั่งของภาษา PHP ส่วนใดคือจุดเริ่มต้น และจุดสิ้นสุดของสคริปต์ PHP ที่อยู่ภายในเอกสาร HTML การแทรกสคริปต์ PHP ลงในเอกสาร HTML นั้น สามารถแทรกได้ 4 รูปแบบ ดังต่อไปนี้

2.1.1 XML Style หรือ Default Syntax

รูปแบบ

```
<?php
        คำสั่งที่ 1;
        คำสั่งที่ 2;
            ...
        คำสั่งที่ N;
?>
```

**ตัวอย่างที่ 2.1** การแทรกสคริปต์ PHP แบบ XML Style

```
1   <?php
2           echo "Hello  World ! <br>";
3           echo "I am  PHP";
4   ?>
```

จากตัวอย่างที่ 2.1 จุดเริ่มต้นของสคริปต์ PHP คือ <?php แล้วตามด้วยชุดคำสั่งที่ต้องการ และจุดสิ้นสุดของสคริปต์ PHP คือ ?> หากไม่ระบุจุดเริ่มต้นและจุดสิ้นสุด โปรแกรมจะไม่สามารถทำงานได้

2.1.2 SGML Style หรือ Short Tags

รูปแบบ

```
<?
        คำสั่งที่ 1;
        คำสั่งที่ 2;
            ...
        คำสั่งที่ N;
?>
```

**ตัวอย่างที่ 2.2** การแทรกสคริปต์ PHP แบบ SGML Style

```
1   <?
2           echo "Hello  World ! <br>";
3           echo "I am  PHP";
4   ?>
```

จากตัวอย่างที่ 2.2 จุดเริ่มต้นของสคริปต์ PHP คือ <? แล้วตามด้วยชุดคำสั่งที่ต้องการ และจุดสิ้นสุดของสคริปต์ PHP คือ ?>

2.1.3 Script Style

รูปแบบ

```
<script language="php">
        คำสั่งที่ 1;
        คำสั่งที่ 2;
            ...
        คำสั่งที่ N;
</script>
```

**ตัวอย่างที่ 2.3** การแทรกสคริปต์ PHP แบบ Script Style

```
1   <script  language="php">
2           echo "Hello  World ! <br>";
3           echo "I am  PHP";
4   </script>
```

จากตัวอย่างที่ 2.3 จุดเริ่มต้นของสคริปต์ PHP คือ <script language="php"> แล้วตามด้วยชุดคำสั่งที่ต้องการ และจุดสิ้นสุดของสคริปต์ PHP คือ </script>

2.1.4 ASP Style

รูปบบ

```
<%
        คำสั่งที่ 1;
        คำสั่งที่ 2;
            ...
        คำสั่งที่ N;
%>
```

**ตัวอย่างที่ 2.4** การแทรกสคริปต์ PHP แบบ ASP Style

```
1   <%
2           echo "Hello  World !  <br>";
3           echo "I am PHP";
4   %>
```

จากตัวอย่างที่ 2.4 จุดเริ่มต้นของสคริปต์ PHP คือ <% แล้วตามด้วยชุดคำสั่งที่ต้องการ และจุดสิ้นสุดของสคริปต์ PHP คือ %>

การแสดงผลลัพธ์ของทั้ง 4 รูปแบบตามตัวอย่างที่ 2.1-2.4 นั้นจะมีผลลัพธ์เหมือนกัน อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 แสดงผลคำว่า Hello World !

บรรทัดที่ 3 แสดงผลคำว่า I am PHP

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

เมื่อ <br> เป็น HTML tag ใช้สำหรับขึ้นบรรทัดใหม่ ส่วน echo เป็นคำสั่งของภาษา PHP ใช้สำหรับการแสดงผลข้อมูลบนเว็บบราวเซอร์ (รายละเอียดการใช้งานคำสั่ง echo จะแนะนำเป็นลำดับต่อไป) หรืออาจจะแทรกสลับไปมาระหว่างสคริปต์ PHP กับ HTML tags ดังตัวอย่างที่ 2.5

**ตัวอย่างที่ 2.5** การเรียกใช้สคริปต์ PHP ร่วมกับ HTML tags

```
1           <html>
2               <head>
3                   <title>My Homepage</title>
4               </head>
5               <body>
```

```
6                    <?php
7                        echo "Hello  World ! <br>";
8                        echo "I am  PHP";
9                    ?>
10               </body>
11           </html>
```

การแทรกสคริปต์ PHP ในเอกสาร HTML ในหนังสือเล่มนี้ทั้งเล่มจะใช้รูปแบบ XML style หรือ Default Syntax เนื่องจากสามารถทำงานได้กับทุกเว็บเซิร์ฟเวอร์ ง่ายต่อการอ่านสคริปต์ และสอดคล้องกับไวยากรณ์ของภาษา XML

## 2.2 องค์ประกอบพื้นฐานของการเขียนโปรแกรมด้วยภาษา PHP

ภาษา PHP นั้นจะใช้โครงสร้างทางภาษาในรูปแบบเดียวกับภาษา C ดังนั้นแนวทางในการเขียนจึงมีลักษณะคล้ายคลึงกัน ทั้งนี้มีองค์ประกอบพื้นฐานบางส่วนที่ควรศึกษาและทำความเข้าใจ เพื่อเป็นพื้นฐานที่ดีสำหรับการเขียนและพัฒนาเว็บเพจต่อไป ดังนี้

### 2.2.1 เครื่องหมายสิ้นสุดคำสั่ง

ภาษา PHP จะใช้เครื่องหมาย ; (Semicolon) เป็นตัวแสดงจุดสิ้นสุดในแต่ละคำสั่ง เช่น

```
$x = 10;
$z = "abc";
```

หลังใส่เครื่องหมาย ; เพื่อสิ้นสุดในแต่ละประโยคคำสั่งแล้ว สามารถนำคำสั่งอื่นมาต่อท้ายได้ทันที แต่การเขียนสคริปต์ลักษณะนี้จะอ่านสคริปต์ได้ยาก ไม่นิยมเขียนในรูปแบบดังกล่าว ตัวอย่างเช่น

```
$x = 10; $y = x + 10 :  $z = "abc";
```

### 2.2.2 คำอธิบาย (Comment)

คำอธิบาย ใช้ในการอธิบายเกี่ยวกับสคริปต์ที่เขียน เพื่อช่วยให้อ่านสคริปต์ได้ง่ายขึ้น แต่โปรแกรมจะไม่นำส่วนที่เป็นคำอธิบายไปประมวลผล สามารถเขียนคำอธิบายได้ 2 รูปแบบ มีรายละเอียด ดังต่อไปนี้

1) Single-Line C++ Syntax หรือ Shell Syntax

เป็นการเขียนคำอธิบายบรรทัดเดียว โดยใช้เครื่องหมาย // หรือ # โปรแกรมจะถือว่าตั้งแต่สัญลักษณ์เป็นต้นไปที่ระบุเป็นคำอธิบายสคริปต์ และจะไม่นำบรรทัดที่ระบุนั้นไปประมวลผล ดังนี้

**ตัวอย่างที่ 2.6** การใช้เครื่องหมาย // หรือ # เพื่อเขียนคำอธิบายแบบบรรทัดเดียว

```
//Author : Parinya Noidonprai
#All Rights Reserved
```

2) Multiple-Line C Syntax

เป็นการเขียนคำอธิบายแบบหลายๆ บรรทัด โดยใช้สัญลักษณ์ /* */ โปรแกรมจะถือว่าตั้งแต่สัญลักษณ์ /* เป็นต้นไปเป็นคำอธิบาย จนกว่าจะเจอสัญลักษณ์ */ จึงจะถือว่าสิ้นสุดคำอธิบาย ตัวอย่าง ดังนี้

**ตัวอย่างที่ 2.7** การใช้เครื่องหมาย /* */ เพื่อเขียนคำอธิบายแบบหลายบรรทัด

```
/* Author : Parinya Noidonprai
   All Rights Reserved */
```

## 2.3 การแสดงผลข้อมูลบนเว็บบราวเซอร์

การแสดงผลข้อมูลบนเว็บบราวเซอร์ คือ การประมวลผลของเว็บเซิร์ฟเวอร์ร่วมกับ Engine ของภาษา PHP แล้วส่งผลลัพธ์กลับเป็นภาษา HTML ไปแสดงผลที่เว็บบราวเซอร์ที่ร้องขอเอกสาร PHP สำหรับในภาษา PHP มี 4 คำสั่งที่สำคัญ คือ print, echo, printf และ sprintf มีรายละเอียด ดังต่อไปนี้

### 2.3.1 คำสั่ง print

คำสั่ง print เป็นคำสั่งที่ใช้แสดงผลลัพธ์ของข้อมูลบนเว็บบราวเซอร์ โดยมีรูปแบบ (คำสั่ง print ไม่ใช่ฟังก์ชันแต่เป็นคำสั่งโครงสร้างพื้นฐานของภาษา PHP) มีรูปแบบและตัวอย่างการใช้งานดังนี้

รูปแบบ

```
int print (string $arg)
```

เมื่อ $arg หมายถึง ตัวแปรข้อมูลนำเข้า (The input data)

หากการทำงานของคำสั่ง print ถูกต้อง จะมีการส่งค่ากลับเป็น "1" เสมอ หากการทำงานไม่ถูกต้องก็จะส่งค่ากลับเป็นค่าอื่นๆ

**ตัวอย่างที่ 2.8** การใช้งานคำสั่ง print

```
1   <?php
2       print ("I love the summertime.");
3   ?>
```

จากตัวอย่างที่ 2.8 เป็นตัวอย่างการใช้งานคำสั่ง print แบบปกติ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 แสดงผลคำว่า I love the summertime.

บรรทัดที่ 3 สิ้นสุดสคริปต์ PHP

**ตัวอย่างที่ 2.9** การแสดงค่าของตัวแปรผ่านคำสั่ง print

```
1   <?php
2       $season = "summertime";
3       print "I love the $season.";
```

```
4       ?>
```

จากตัวอย่างที่ 2.9 เป็นตัวอย่างการใช้คำสั่ง print โดยแสดงผลค่าของตัวแปร อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $season มีค่าเท่ากับ summertime

บรรทัดที่ 3 แสดงผลคำว่า I love the summertime. คำว่า summertime มาจากค่าที่เก็บไว้ในตัวแปร $season

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

**ตัวอย่างที่ 2.10** การใช้ประโยคพารามิเตอร์ (Parameter) ที่อยู่คนละบรรทัดผ่านคำสั่ง print

```
1       <?php
2               print "I love the
3               summertime.";
4       ?>
```

จากตัวอย่างที่ 2.10 ตัวอย่างการใช้ประโยคพารามิเตอร์ ที่อยู่คนละบรรทัดผ่านคำสั่ง print ก็สามารถแสดงผลลัพธ์ได้เหมือนกัน เพราะคำสั่ง print ออกแบบมาให้ง่ายและยืดหยุ่น ดังนั้นจะใส่วงเล็บหรือไม่ก็ได้ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 แสดงผลคำว่า I love the

บรรทัดที่ 3 แสดงผลคำว่า summertime. ต่อกับบรรทัดก่อนหน้านี้

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

2.3.2 คำสั่ง echo

คำสั่ง echo เป็นคำสั่งที่ใช้แสดงผลลัพธ์ของข้อมูลบนเว็บบราวเซอร์ เหมือนกับคำสั่ง print ความแตกต่าง คือ คำสั่ง print จะมีการตรวจสอบข้อผิดพลาดมากกว่าคำสั่ง echo โดยรูปแบบของคำสั่ง echo (คำสั่ง echo ไม่ใช่ฟังก์ชันแต่เป็นคำสั่งโครงสร้างพื้นฐานของภาษาเช่นเดียวกับคำสั่ง print ) มีรูปแบบและตัวอย่างการใช้งานดังนี้

รูปแบบ

```
        void echo ( string $arg1 [, string $... ] )
```

เมื่อ $arg1 หมายถึง ตัวแปรข้อมูลนำเข้า (สามารถกำหนดได้หลายตัวแปร)

**ตัวอย่างที่ 2.11** การใช้คำสั่ง echo

```
1       <?php
2               $short = "SRU";
3               $long = "Suratthani Rajabhat University.";
4               echo $short, " is ", $long, " We are a great university.";
```

```
5       ?>
```

ผลลัพธ์

```
SRU is Suratthani Rajabhat University. We are a great university.
```

จากตัวอย่างที่ 2.11 ตัวอย่างการใช้คำสั่ง echo อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $short มีค่าเท่ากับ SRU

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $long มีค่าเท่ากับ Suratthani Rajabhat University.

บรรทัดที่ 4 แสดงผลคำว่า SRU is Suratthani Rajabhat University. We are a great university. เป็นการแสดงผลร่วมกันระหว่างข้อมูลแบบคงที่และข้อมูลแบบไดนามิกผ่านตัวแปร

บรรทัดที่ 5 สิ้นสุดสคริปต์ PHP

คำแนะนำเพิ่มเติม หากต้องการแสดงผลลัพธ์ที่มีการผสมผสานระหว่างข้อมูลแบบคงที่และข้อมูลแบบไดนามิกผ่านตัวแปร ขอให้พิจารณาใช้ printf แทน ซึ่งจะแนะนำในลำดับต่อไป หากเป็นเพียงข้อความแบบคงที่ธรรมดาทั่วๆ ไป ขอแนะนำให้ใช้คำสั่ง echo หรือ print เนื่องจากง่ายต่อการใช้งาน

2.3.3 คำสั่ง printf

คำสั่ง printf ใช้สำหรับแสดงผลลัพธ์ในรูปแบบข้อความ (Formatted String) เหมาะสำหรับการแสดงผลลัพธ์แบบผสมผสานระหว่างข้อความแบบคงที่และข้อมูลแบบไดนามิกที่เก็บไว้ในตัวแปรหรือหลายๆ ตัวแปร และยังสามารถควบคุมการแสดงผลข้อมูลแบบไดนามิก มีรูปแบบและตัวอย่างการใช้งานดังนี้

รูปแบบ

```
int printf ( string $format [, mixed $args [, mixed $... ]] )
```

เมื่อ $format หมายถึง ตัวกำหนดชนิดการแสดงผล (Type Specifies)

$args หมายถึง ตัวแปรข้อมูลนำเข้า

**ตัวอย่างที่ 2.12** การใช้คำสั่ง printf สำหรับแสดงผลลัพธ์ในรูปแบบข้อความ

```
1       <?php
2           printf ("Bar inventory: %d bottles of tonic water.", 100);
3       ?>
```

ผลลัพธ์

```
Bar inventory: 100 bottles of tonic water.
```

จากตัวอย่างที่ 2.12 แสดงตัวอย่างการใช้คำสั่ง printf สำหรับแสดงผลลัพธ์ในรูปแบบข้อความ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 แสดงผลลัพธ์ในรูปแบบข้อความด้วยคำสั่ง printf ในตัวอย่างมีการกำหนดรูปแบบการแสดงผลด้วย %d คือ ตัวกำหนดชนิดการแสดงผลลัพธ์เป็นค่าของเลขจำนวนเต็ม เมื่อใช้คำสั่ง printf มีค่าพารามิเตอร์ ที่ส่งเข้ามา คือ 100 ค่า 100 จะถูกแทนที่ใน %d ระหว่างข้อความ กรณีที่ ค่าของตัวเลขที่ส่งเข้ามาเป็นค่าทศนิยม ค่าดังกล่าวจะถูกปัดเศษให้เป็นจำนวนเต็มที่ใกล้เคียงที่สุด เช่น ถ้าส่งค่าเป็น 100.2 ผลลัพธ์จะแสดงเป็น 100 สำหรับตัวกำหนดชนิดการแสดงผล มีรายละเอียดดังตารางที่ 2.1

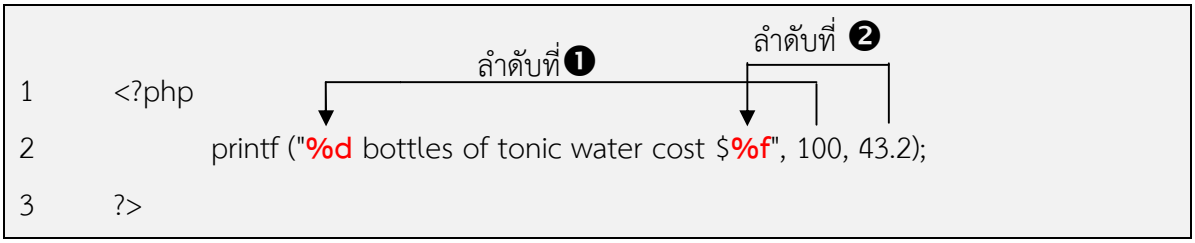
บรรทัดที่ 3 สิ้นสุดสคริปต์ PHP

**ตารางที่ 2.1** แสดงตัวกำหนดชนิดการแสดงผลที่ใช้ร่วมกับคำสั่ง printf

| ตัวกำหนดชนิด | ความหมาย |
|---|---|
| %b | เป็นชนิดตัวเลข แสดงผลเป็นเลขไบนารี (a binary number) |
| %c | เป็นชนิดตัวเลข แสดงผลเป็นตัวอักษร ASCII |
| %d | เป็นชนิดเลขจำนวนเต็มบวกและลบ แสดงผลเป็นตัวเลขฐานสิบ |
| %f | เป็นชนิดเลขจำนวนจริง แสดงผลเป็นตัวเลขฐานสิบและทศนิยม |
| %o | เป็นชนิดเลขจำนวนเต็ม แสดงผลเป็นตัวเลขฐานแปด |
| %s | เป็นชนิดข้อความ แสดงผลเป็นข้อความ |
| %u | เป็นชนิดเลขจำนวนเต็มบวก แสดงผลเป็นตัวเลขฐานสิบ |
| %x | เป็นชนิดตัวเลข แสดงผลเป็นเลขฐานสิบหก (อักษรพิมพ์เล็ก) |
| %X | เป็นชนิดตัวเลข แสดงผลเป็นเลขฐานสิบหก (อักษรพิมพ์ใหญ่) |

ดังนั้นหากต้องการที่จะส่งผ่านข้อมูล 2 ค่า สามารถทำได้โดยระบุตัวกำหนดชนิด 2 ตำแหน่ง โดยค่าที่ถูกส่งจะเรียงตามลำดับก่อนหลัง ดังตัวอย่างที่ 2.13

**ตัวอย่างที่ 2.13** ตัวอย่างการใช้คำสั่ง printf โดยระบุตัวกำหนดชนิดมากกว่า 1 ชนิด

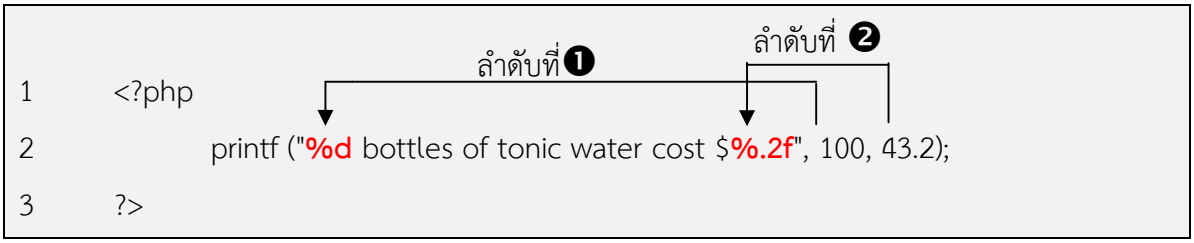



ผลลัพธ์

```
100 bottles of tonic water cost $43.200000
```

หากต้องการกำหนดจำนวนของการแสดงผลจำนวนเลขทศนิยม สามารถกำหนดได้โดยระบุตัวเลขไว้ก่อนหน้าตัวกำหนดชนิด เช่น หากต้องการกำหนดรูปแบบให้แสดงจำนวนเลขทศนิยมจำนวน 2 หลัก จำเป็นต้องใช้ตัวกำหนดชนิด %f เนื่องจากเป็นเลขจำนวนจริงพร้อมระบุจำนวนเลขที่ต้องการ ดังนี้

**ตัวอย่างที่ 2.14** ตัวอย่างการใช้คำสั่ง printf เพื่อการแสดงผลจำนวนเลขทศนิยม




```
1    <?php
2        printf ("%d bottles of tonic water cost $%.2f", 100, 43.2);
3    ?>
```

ผลลัพธ์

```
100 bottles of tonic water cost $43.20
```

จากตัวอย่างที่ 2.13 และ 2.14 แสดงตัวอย่างการใช้คำสั่ง printf อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 แสดงผลลัพธ์ในรูปแบบข้อความด้วยคำสั่ง printf ในตัวอย่างมีการกำหนดรูปแบบการแสดงผลตำแหน่งที่ 1 ด้วย %d คำสั่งจะส่งค่าข้อมูลมาให้ตำแหน่ง %d ด้วยค่า 100 แล้วตามด้วยข้อความ bottles of tonic water cost $ และตำแหน่งที่ 2 กำหนดรูปแบบด้วย %f (ทั้ง 2 ตัวอย่างต่างกันที่ ตัวอย่างที่ 2.14 ไม่ได้ระบุจำนวนหลักทศนิยม) คำสั่งจะส่งค่าข้อมูลมาให้ตำแหน่ง %f หรือ %.2f ด้วยค่า 43.2

บรรทัดที่ 3 สิ้นสุดสคริปต์ PHP

2.3.4 คำสั่ง sprintf

คำสั่ง sprintf เป็นคำสั่งที่ทำหน้าที่เหมือนคำสั่ง printf ข้อแตกต่าง คือ sprintf ใช้สำหรับส่งค่ากลับเป็นข้อความหรือเป็นการนำข้อความมาต่อให้เป็นประโยคเดียวกันแล้วกำหนดค่าให้กับตัวแปร เพื่อประโยชน์การทำงานตามเหมาะสม ไม่มีการแสดงผลออกไปยังเว็บบราวเซอร์ ลักษณะการใช้งานเหมือน printf ดังนั้นจะใช้ ตัวกำหนดชนิด เหมือนกัน ตามตารางที่ 2.1 มีรูปแบบและตัวอย่างการใช้งานดังนี้

รูปแบบ

```
string sprintf ( string $format [, mixed $args [, mixed $... ]] )
```

**ตัวอย่างที่ 2.15** การใช้คำสั่ง sprintf สำหรับส่งค่ากลับเป็นข้อความให้เป็นประโยคเดียวกัน

```
1    <?php
2        $text = sprintf ("%d bottles of tonic water cost $%.2f", 100, 43.2);
3        echo $text;
4    ?>
```

ผลลัพธ์

```
100 bottles of tonic water cost $43.20
```

จากตัวอย่างที่ 2.15 แสดงตัวอย่างการใช้คำสั่ง sprintf อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $text มีค่าเท่ากับ 100 bottles of tonic water cost $ 43.20 โดย 100 ถูกส่งมาให้กับตำแหน่ง %d และ 43.20 ถูกส่งมาให้กับตำแหน่ง %.2f

บรรทัดที่ 3 แสดงผลค่าข้อมูลตัวแปร $text

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

## 2.4 ชนิดข้อมูลที่รองรับในภาษา PHP

ในภาษา PHP มีข้อมูลอยู่หลายชนิด ไม่ว่าจะเป็นข้อมูลชนิดบูลีน (Boolean) เลขจำนวนเต็ม (Integer) เลขจำนวนจริง (Float) ข้อความ (String) และอาร์เรย์ (Array) เพื่อทำความเข้าใจและสามารถนำไปใช้ร่วมกับการกำหนดตัวแปรในหัวข้อต่อๆ ไป ดังรายละเอียดต่อไปนี้

### 2.4.1 ชนิดข้อมูลพื้นฐาน (Scalar Data Types)

ชนิดข้อมูลพื้นฐาน เป็นชนิดข้อมูลที่กำหนดให้ตัวแปรหนึ่งๆ เก็บข้อมูลได้เพียงค่าเดียว ชนิดของข้อมูลที่จัดอยู่ในกลุ่มนี้ประกอบด้วย บูลีน เลขจำนวนเต็ม เลขจำนวนจริง และข้อความ มีรายละเอียด ดังนี้

1) ข้อมูลชนิดบูลีน

ข้อมูลชนิดบูลีนเป็นชื่อหลังของ George Boole (1815-1864) นักคณิตศาสตร์ที่ได้รับการยกย่องให้เป็นหนึ่งในผู้ก่อตั้งทฤษฎีข้อมูลตัวแปรบูลีน เป็นตัวแทนของความจริงที่สนับสนุนเพียงสองค่าเท่านั้น คือ true (จริง) และ false (เท็จ) หรือสามารถใช้เลขศูนย์แทน false และค่าใดๆ ที่ไม่ใช่ศูนย์แทนค่าของ true ดังตัวอย่างต่อไปนี้

**ตัวอย่างที่ 2.16** การกำหนดค่าตัวแปรชนิดบูลีน

```
1    <?php
2        $alive = false;
3        $alive = true;
4        $alive = 1;
5        $alive = 0;
6    ?>
```

จากตัวอย่างที่ 2.16 แสดงตัวอย่างการกำหนดค่าตัวแปรชนิดบูลีน อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $alive มีค่าเป็น false (มีค่าเป็นเท็จ)

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $alive มีค่าเป็น true (มีค่าเป็นจริง)

บรรทัดที่ 4 กำหนดค่าให้กับตัวแปร $alive มีค่าเป็น 1 (มีค่าเป็นจริง)

บรรทัดที่ 5 กำหนดค่าให้กับตัวแปร $alive มีค่าเป็น 0 (มีค่าเป็นเท็จ)

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

นอกจากการกำหนดตัวแปรเป็นชนิดบูลีนแล้ว การได้มาของข้อมูลชนิดบูลีนยังได้มาจากการเปรียบเทียบทางตรรกะ เพื่อตัดสินใจดำเนินการร่วมกับโครงสร้างควบคุมอื่นๆ ที่เกี่ยวข้อง เช่น หากผลลัพธ์ออกมาเป็น true ก็จะทำงานต่อ แต่ถ้าออกมาเป็น false ก็จะหยุดทำงาน เป็นต้น

2) ข้อมูลชนิดเลขจำนวนเต็ม

ข้อมูลชนิดเลขจำนวนเต็มเป็นตัวแทนของจำนวนเต็มใดๆ ที่ไม่ประกอบด้วยส่วนที่เป็นเศษส่วน ภาษา PHP สนับสนุนค่าจำนวนเต็มแสดงในเลขฐาน 10 (Decimal) เลขฐาน 8 (Octal) และเลขฐาน 16 (Hexadecimal) ตัวอย่าง ดังนี้

**ตัวอย่างที่ 2.17** ตัวอย่างข้อมูลชนิดเลขจำนวนเต็ม

```
42      // เลขฐาน 10
0755    // เลขฐาน 8
0xC4E   // เลขฐาน 16
```

ขนาดสูงสุดของข้อมูลชนิดเลขจำนวนเต็มสูงสุดที่รองรับ ขึ้นอยู่กับระบบปฏิบัติการที่ใช้ สำหรับ PHP 5 และเวอร์ชันก่อนหน้านี้ จะเป็น 32 bit จะมีขนาดสูงสุดของเลขเต็มบวกและเต็มลบเท่ากับ $2^{31}$ สำหรับ PHP 6 จะสนับสนุนค่าสูงสุดของเลขเต็มบวกและเต็มลบเท่ากับ $2^{63}$

3) ข้อมูลชนิดเลขจำนวนจริง

ข้อมูลชนิดเลขจำนวนจริงหรือเรียกว่าเลขที่มีทศนิยม ช่วยให้สามารถระบุตัวเลขที่มีรายละเอียดที่แท้จริงของเลข เช่น ใช้แทนค่าทางการเงิน น้ำหนัก ระยะทาง และพื้นที่ทั้งหมด เป็นต้น มีตัวอย่าง ดังนี้

**ตัวอย่างที่ 2.18** แสดงตัวอย่างข้อมูลชนิดเลขจำนวนจริง

```
4.5678
4.0
8.7e4
1.23E+11
```

4) ข้อมูลชนิดข้อความ

ข้อมูลชนิดข้อความ คือ ข้อมูลที่เป็นอักษร อาจจะเป็นภาษาอังกฤษ ไทย อื่นๆ ก็ได้ แต่เนื่องจากข้อความมีความยาวเท่าไหร่ก็ได้ และไม่จำเป็นต้องเขียนอักษรทุกตัวติดกันไปจนจบข้อความ เหมือนกับการเขียนตัวเลข จึงทำให้โปรแกรมไม่สามารถตัดสินใจได้ว่า ข้อความนั้นเริ่มต้นและสิ้นสุดที่ใด ดังนั้นการกำหนดข้อมูลที่เป็นข้อความ ต้องกำหนดจุดเริ่มต้นและสิ้นสุดเสมอ ด้วยเครื่องหมาย Double Quotes ("...") หรือ Single Quotes ('... ') เช่น "Hello, world" หรือ 'Welcome to Thailand' เป็นต้น แต่อย่างไรก็ตาม การใช้เครื่องหมาย Quotes ทั้งสองแบบจะมีข้อแตกต่างกันในบางกรณี ซึ่งจะได้กล่าวถึงในลำดับต่อไป

**ตัวอย่างที่ 2.19** แสดงตัวอย่างข้อมูลชนิดข้อความ

```
"PHP is a great language"
"Suratthani-Rajabhat-University"
'*9subway\n'
```

นอกจากนี้ยังสามารถเลือกอักษรบางตัวจากข้อความทั้งหมดได้ โดยใช้อาร์เรย์ ดังนี้

**ตัวอย่างที่ 2.20** ตัวอย่างการเลือกอักษรบางตัวจากข้อความทั้งหมด

```
1   <?php
2       $name = "PARINYA";
3       $parser = $name[2];
4       echo $parser;
5   ?>
```

ผลลัพธ์

```
R
```

จากตัวอย่างที่ 2.20 ตัวอย่างการเลือกอักษรบางตัวจากข้อความทั้งหมด อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $name มีค่าเท่ากับ PARINYA

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $parser มีค่าเท่ากับอักษรจากข้อความที่อยู่ในตัวแปร $name ตำแหน่งอักษรที่ 2 เมื่อ $name มีรูปแบบการเก็บสายอักษร คือ P[0] A[1] R[2] I[3] N[4] Y[5] และ A[6] ดังนั้น เมื่อกำหนดให้ $parser=$name[2] จึงหมายความว่า กำหนดค่าให้ตัวแปร $parser มีค่าเท่ากับ R นั่นเอง (วิธีการเรียงลำดับเริ่มจากซ้ายมือ เริ่มต้นด้วยลำดับเลข 0)

บรรทัดที่ 4 แสดงค่าตัวแปร $parser คือ R

บรรทัดที่ 5 สิ้นสุดสคริปต์ PHP

2.4.2 ชนิดข้อมูลแบบชุด (Compound Data Types)

ชนิดข้อมูลแบบชุด เป็นการรวมชุดของรายการที่คล้ายกันไว้ด้วยกัน อาจเป็นตัวแปรชื่อเดียวกันเป็นข้อมูลชนิดเดียวกันเพื่อใช้ในการอ้างอิงถึงกลุ่มข้อมูลชนิดเดียวกัน มีรายละเอียด ดังนี้

1) ชนิดข้อมูลอาร์เรย์ (Array Data Types)

อาร์เรย์จะมีประโยชน์ในการรวมชุดของรายการที่คล้ายกันรวมกันไว้ด้วยกัน เพื่อใช้สำหรับการอ้างอิงถึงข้อมูลอะไรบางอย่างเฉพาะเจาะจง โครงสร้างข้อมูลที่เรียกว่าอาร์เรย์ ที่ถูกกำหนดอย่างมีรูปแบบเป็นกลุ่มเดียวกัน มีดัชนีเพื่อใช้เข้าถึงหรืออ้างถึงสมาชิกข้อมูลในลำดับต่างๆ การอ้างถึงสมาชิกหนึ่งๆ ในอาร์เรย์จะต้องใช้ดัชนีหรือคีย์ของอาร์เรย์ในการชี้ตำแหน่ง อ้างอิงค่าที่สอดคล้องกัน และแต่ละข้อมูลในสมาชิกสามารถที่จะอ้างอิงด้วยตัวเลขที่เรียบง่ายไปยังตำแหน่งของค่าในชุดหรืออาจมีบางส่วนสัมพันธ์โดยตรงกับค่า

**ตัวอย่างที่ 2.21** ตัวอย่างการกำหนดค่าสมาชิกให้กับตัวแปรชนิดอาร์เรย์โดยใช้ดัชนีอาร์เรย์

```
<?php
    $countries [0] = "Thailand";
    $countries [1] = "Myanmar";
        ...
    $countries [50] = "Cambodia";
?>
```

นอกจากนี้ยังสามารถสร้างดัชนีความสัมพันธ์เพื่อเชื่อมโยงข้อมูล เช่น สร้างความสัมพันธ์ของอาร์เรย์ไปยังเมืองหลวงของประเทศต่างๆ แทนที่จะใช้ดัชนีที่เป็นตัวเลข โดยใช้ชื่อของประเทศแทนดัชนีลักษณะเช่นนี้จะเรียกว่าคีย์อาร์เรย์สำหรับสร้างความเชื่อมโยงแทน ดังนี้

**ตัวอย่างที่ 2.22** แสดงตัวอย่างการกำหนดค่าสมาชิกให้กับตัวแปรชนิดอาร์เรย์โดยใช้คีย์อาร์เรย์

```
<?php
    $capital ["Thailand"] = "Bangkok";
    $capital ["Myanmar"] = "Naypyidaw";
        ...
    $capital ["Cambodia"] = "Phnom Penh";
?>
```

2) ชนิดข้อมูลวัตถุ (Object Data Types)

ชนิดข้อมูลวัตถุ ที่สนับสนุนในภาษา PHP มีแนวคิดมาจากกระบวนทัศน์ในการเขียนโปรแกรมเชิงวัตถุ แตกต่างจากชนิดข้อมูลอื่นๆ ในภาษา PHP วัตถุจะต้องประกาศอย่างชัดเจน การประกาศลักษณะของวัตถุ และลักษณะการทำงานนี้จะเกิดขึ้นภายในสิ่งที่เรียกว่า "คลาส" มีตัวอย่าง ดังนี้

**ตัวอย่างที่ 2.23** แสดงตัวอย่างการเขียนโปรแกรมเชิงวัตถุด้วยภาษา PHP

```
1  <?php
2      class Appliance {
3          private $_power;
4          function setPower ($status) {
5              $this->_power = $status;      // การเรียกใช้เมธอดภายในคลาสเดียวกัน
6          }
7      }
8      $blender = new Appliance;             // การกำหนดให้ $blender ให้สามารถ
9                                            // เรียกใช้เมธอดของคลาส Appliance
10 ?>
```

## 2.5 การแปลงชนิดข้อมูล

การแปลงชนิดข้อมูลหนึ่งไปเป็นชนิดข้อมูลอื่นๆ ได้นั้น จำเป็นต้องรู้ถึงชนิดของข้อมูลที่จะแปลงไป การแปลงค่าชนิดข้อมูลสามารถทำได้โดยการวางชนิดที่ต้องการแปลงไว้ในด้านหน้าตัวแปรที่จะทำการแปลง สำหรับรายละเอียดชนิดของตัวแปลง แสดงในตารางที่ 2.2 มีรายละเอียด ดังต่อไปนี้

**ตารางที่ 2.2** ตัวดำเนินการแปลงชนิดของข้อมูล

| Cast Operators | Conversion | ความหมาย |
|---|---|---|
| (array) | Array | กำหนดให้เป็นข้อมูลชนิดอาร์เรย์ |
| (bool) or (boolean) | Boolean | กำหนดให้เป็นข้อมูลชนิดบูลีน |
| (int) or (integer) | Integer | กำหนดให้เป็นข้อมูลชนิดเลขจำนวนเต็ม |
| (int64) 64-bit | integer | กำหนดให้เป็นข้อมูลชนิดเลขจำนวนเต็ม 64 บิต |
| (object) | Object | กำหนดให้เป็นข้อมูลชนิดวัตถุ |
| (real) or (double) or (float) | Float | กำหนดให้เป็นข้อมูลชนิดเลขจำนวนจริง |
| (string) | String | กำหนดให้เป็นข้อมูลชนิดข้อความ |

**ตัวอย่างที่ 2.24** การแปลงเลขจำนวนเต็มเป็นเลขจำนวนจริง

```
1    <?php
2        $score=(double)13;
3        printf("Data Type is: %s = %.2f", gettype($score), $score);
4    ?>
```

ผลลัพธ์

```
Data Type is: double = 13.00
```

จากตัวอย่างที่ 2.24 การแปลงเลขจำนวนเต็มเป็นเลขจำนวนจริง อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดชนิดให้กับตัวแปร $score เป็นชนิดเลขจำนวนจริง และกำหนดค่าให้มีค่าเท่ากับค่าที่ได้มาจากการแปลงเลขจำนวนเต็มเป็นเลขจำนวนจริง ด้วย (double)13

บรรทัดที่ 3 ใช้คำสั่ง printf เพื่อแสดงผลข้อมูลจึงได้ผลลัพธ์ตามตัวอย่าง (เมื่อฟังก์ชัน gettype ใช้สำหรับดูชนิดของข้อมูลว่าเป็นชนิดอะไร ผลของฟังก์ชันจะส่งผลกลับเป็นข้อมูลชนิดข้อความ ดังนั้นภายในคำสั่ง printf จึงต้องกำหนดรูปแบบการแสดงผลด้วย %s)

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

ตัวอย่างการแปลงค่าข้อมูลชนิดเลขจำนวนจริง เป็นเลขจำนวนเต็ม ในกรณีที่ค่าของข้อมูลมีเลขทศนิยม เลขทศนิยมที่อยู่ด้านหลังจะถูกปัดทิ้ง ตัวอย่างดังนี้

**ตัวอย่างที่ 2.25** การแปลงค่าข้อมูลชนิดเลขจำนวนจริงเป็นเลขจำนวนเต็ม

```
1    <?php
2        $score=(int)13.99;
3        printf("Data Type is: %s = %d", gettype($score), $score);
4    ?>
```

ผลลัพธ์

```
Data Type is: integer = 13
```

จากตัวอย่างที่ 2.25 การแปลงค่าข้อมูลชนิดเลขจำนวนจริงเป็นเลขจำนวนเต็ม อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดชนิดให้กับตัวแปร $score เป็นชนิดเลขจำนวนเต็ม และกำหนดค่าให้เท่ากับค่าที่ได้มาจากการแปลงเลขจำนวนจริงเป็นเลขจำนวนเต็ม ด้วย (int)13.99

บรรทัดที่ 3 ใช้คำสั่ง printf เพื่อแสดงผลข้อมูลจึงได้ผลลัพธ์ตามตัวอย่าง โดยเลขทศนิยมที่อยู่ด้านหลังจะถูกปัดทิ้งทั้งหมด

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

**ตัวอย่างที่ 2.26** การแปลงค่าข้อมูลชนิดข้อความไปเป็นชนิดเลขจำนวนเต็ม

```
1    <?php
2        $score = (int)"This is a text of 2015";
3        printf("Data Type is: %s = %d",gettype($score),$score);
4    ?>
```

ผลลัพธ์

```
Data Type is: integer = 0
```

จากตัวอย่างที่ 2.26 การแปลงค่าข้อมูลชนิดข้อความไปเป็นชนิดเลขจำนวนเต็ม อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดชนิดให้กับตัวแปร $score เป็นชนิดเลขจำนวนเต็ม และกำหนดค่าให้มีเท่ากับค่าที่ได้มาจากการแปลงข้อความเป็นเลขจำนวนเต็ม ด้วย (int)"This is a text of 2015"

บรรทัดที่ 3 ใช้คำสั่ง printf เพื่อแสดงผลข้อมูลจึงได้ผลลัพธ์ตามตัวอย่าง จะเห็นได้ว่าข้อมูลชนิดข้อความเมื่อถูกแปลงข้อมูลเป็นชนิดตัวเลข ไม่ว่าจะเป็นเลขจำนวนเต็มหรือเลขจำนวนจริง ค่าของตัวแปรจะมีค่าเป็น 0 ดังนั้นควรระวัง เมื่อมีการแปลงข้อมูลลักษณะเช่นนี้

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

**ตัวอย่างที่ 2.27** การแปลงค่าข้อมูลปกติเป็นข้อมูลชนิดอาร์เรย์

```
1    <?php
2        $name = "Suratthani Rajabhat University";
3        $arrayname = (array)$name;
4        printf("Data Type is: %s = %s", gettype($arrayname), $arrayname[0]);
5    ?>
```

ผลลัพธ์

```
Data Type is: array = Suratthani Rajabhat University
```

จากตัวอย่างที่ 2.27 การแปลงค่าข้อมูลปกติเป็นข้อมูลชนิดอาร์เรย์ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $name มีค่าเป็นข้อความคำว่า "Suratthani Rajabhat University"

บรรทัดที่ 3 กำหนดชนิดให้กับตัวแปร $arrayname เป็นชนิดอาร์เรย์ และกำหนดค่าให้กับดัชนีอาร์เรย์ลำดับที่ 0 มีค่าเท่ากับค่าที่อยู่ในตัวแปร $name

บรรทัดที่ 4 ใช้คำสั่ง printf เพื่อแสดงผลข้อมูลจึงได้ผลลัพธ์ตามตัวอย่าง

บรรทัดที่ 5 สิ้นสุดสคริปต์ PHP

**ตัวอย่างที่ 2.28** การแปลงค่าข้อมูลปกติเป็นข้อมูลชนิดวัตถุ

```
1    <?php
2        $name = "Suratthani Rajabhat University";
3        $obj = (object) $name;
4        printf("Data Type is: %s = %s", gettype($obj), $obj -> scalar);
5    ?>
```

ผลลัพธ์

```
Data Type is: object = Suratthani Rajabhat University
```

จากตัวอย่างที่ 2.28 การแปลงค่าข้อมูลปกติเป็นข้อมูลชนิดวัตถุ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $name มีค่าเป็นข้อความคำว่า "Suratthani Rajabhat University"

บรรทัดที่ 3 กำหนดชนิดให้กับตัวแปร $obj เป็นชนิดวัตถุ และกำหนดค่าให้เท่ากับค่าที่อยู่ในตัวแปร $name

บรรทัดที่ 4 ใช้คำสั่ง printf แสดงค่าของตัวแปรวัตถุด้วยรูปแบบ $obj -> scalar เพื่อแสดงผลค่าข้อมูลจึงได้ผลลัพธ์ตามตัวอย่าง

บรรทัดที่ 5 สิ้นสุดสคริปต์ PHP

## 2.6 การปรับเปลี่ยนชนิดข้อมูลแบบอัตโนมัติ

ภาษา PHP ถูกออกแบบมาให้มีความยืดหยุ่นสูง สามารถปรับเปลี่ยนค่าตัวแปรได้หลากหลายและทันท่วงทีแบบอัตโนมัติ หรือสามารถปรับเปลี่ยนชนิดของตัวแปรแบบอัตโนมัติ โดยระบบจะทำการพิจารณาความเหมาะสมกับสถานการณ์ที่มีการอ้างอิงตัวแปร มีตัวอย่างดังนี้

**ตัวอย่างที่ 2.29** การปรับเปลี่ยนชนิดข้อมูลแบบอัตโนมัติ แบบที่ 1

```
1   <?php
2       $total=5;
3       $count="15";
4       printf("Data Type of total variable is : %s<br>", gettype($total));
5       printf("Data Type of count variable is : %s<br>", gettype($count));
6       printf("%d + %d ", $total, $count);
7       $total=$total+$count;
8       printf(" = %d", $total);
9   ?>
```

ผลลัพธ์

```
Data Type of total variable is : integer
Data Type of count variable is : string
5 + 15 = 20
```

จากตัวอย่างที่ 2.29 การปรับเปลี่ยนชนิดข้อมูลแบบอัตโนมัติ แบบที่ 1 อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดค่าให้กับตัวแปร $total มีค่าเท่ากับ 5 (เลขจำนวนเต็ม)

บรรทัดที่ 3 กำหนดค่าให้กับตัวแปร $count มีค่าเท่ากับ 15 (ข้อความ)

บรรทัดที่ 4 และ 5 แสดงผลชนิดของตัวแปรแต่ละตัว โดยใช้ฟังก์ชัน gettype( )

บรรทัดที่ 6 แสดงค่าตัวแปร $total + $count ผลลัพธ์ที่ได้ คือ 5 + 15

บรรทัดที่ 7 กำหนดค่าใหม่ให้กับตัวแปร $total มีค่าเท่ากับผลรวมของตัวแปร $total+$count คือ มีค่าเท่ากับ 20 (ลักษณะดังกล่าวจึงถือได้ว่าเป็นการปรับเปลี่ยนชนิดข้อมูลแบบอัตโนมัติ ภาษาอื่นๆ จะไม่สามารถนำเอาเลขจำนวนเต็มมาบวกกับข้อความได้)

บรรทัดที่ 8 แสดงเครื่องหมายเท่ากับและค่าตัวแปร $total คือ =20

บรรทัดที่ 9 สิ้นสุดสคริปต์ PHP

**ตัวอย่างที่ 2.30** การปรับเปลี่ยนชนิดข้อมูลแบบอัตโนมัติ แบบที่ 2

```
1   <?php
2       $total = "45 fire engines";
```

```
3           $incoming = 10;
4           $total = $incoming + $total;
5           printf("%d ", $total);
6       ?>
```

ผลลัพธ์

```
55
```

จากตัวอย่างที่ 2.30 การปรับเปลี่ยนชนิดข้อมูลแบบอัตโนมัติ แบบที่ 2 อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $total มีค่าเป็นชนิดข้อความ และกำหนดให้มีค่าเท่ากับ "45 fire engines"

บรรทัดที่ 3 กำหนดให้ตัวแปร $incoming เป็นชนิดเลขจำนวนเต็ม และกำหนดให้มีค่าเท่ากับ 10

บรรทัดที่ 4 กำหนดให้ตัวแปร $total เดิมเป็นชนิดข้อความ เมื่อกำหนดให้มีค่าเท่ากับผลบวกของตัวมันเองบวกด้วยค่าในตัวแปร $incoming โดยระบบจะตรวจสอบว่าข้อความที่อยู่ในตัวแปร $total ขึ้นต้นด้วยตัวเลขหรือไม่ หากมีระบบจะเลือกค่าตัวเลขในช่วงต้นแล้วตัดข้อมูลที่เป็นข้อความออกทั้งหมดแล้วนำไปบวกกับค่าในตัวแปร $incoming โดยสรุปกล่าวคือ จากสมการ $total = $incoming + $total จะเป็นการกำหนดให้ตัวแปร $total มีค่าเท่ากับ 45+10 ผลลัพธ์ที่ได้จึงมีค่าเท่ากับ 55

บรรทัดที่ 5 แสดงผลค่าตัวแปร $total

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

## 2.7 ฟังก์ชันที่เกี่ยวข้องกับชนิดข้อมูล

ฟังก์ชันที่เกี่ยวข้องกับชนิดข้อมูล (Type-Related Functions) มีรายละเอียด ดังนี้

### 2.7.1 ฟังก์ชันเรียกดูชนิดของข้อมูล (Retrieving Types Functions)

ฟังก์ชัน gettype ( ) เป็นฟังก์ชันที่ใช้สำหรับเรียกดูชนิดของข้อมูล โดยการส่งผลกลับเป็นชนิดของข้อมูลในตัวแปร ค่าที่ส่งกลับมีทั้งหมด 9 ค่า ประกอบด้วย boolean, integer, double หรือ float, string, array, object, resource, NULL และ unknown type รูปแบบการใช้งาน ดังนี้

รูปแบบ

```
string gettype ( mixed $var )
```

เมื่อ $var หมายถึง ตัวแปรที่ต้องการตรวจสอบชนิด

**ตัวอย่างที่ 2.31** การใช้ฟังก์ชัน gettype ( ) เรียกดูชนิดของข้อมูล

```
1       <?php
2           $name="Suratthani Rajabhat University";
3           printf("Data Type is: %s ", gettype($name));
4       ?>
```

ผลลัพธ์

```
Data Type is: string
```

จากตัวอย่างที่ 2.31 การใช้ฟังก์ชัน gettype ( ) เรียกดูชนิดของข้อมูล อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $name มีค่าเท่ากับ Suratthani Rajabhat University

บรรทัดที่ 3 แสดงผลชนิดของตัวแปร $name ด้วยใช้ฟังก์ชัน gettype( ) สำหรับเรียกดูชนิดของข้อมูลที่อยู่ในตัวแปร

บรรทัดที่ 4 สิ้นสุดสคริปต์ PHP

2.7.2 ฟังก์ชันแปลงชนิดข้อมูล (Converting Types Functions)

ฟังก์ชัน settype ( ) เป็นฟังก์ชันสำหรับแปลงตัวแปรตามที่ระบุให้เป็นชนิดอื่นๆ ตามต้องการ ชนิดของข้อมูลที่สามารถแปลงได้นั้น มี 7 ชนิด ประกอบด้วย array, boolean, float, integer, null, object และ string นอกจากนี้ถ้าหากการแปลงชนิดข้อมูลประสบความสำเร็จ ฟังก์ชัน settype ( ) จะมีการส่งค่ากลับผลลัพธ์จะมีค่าเป็น TRUE หากไม่สำเร็จจะส่งค่ากลับเป็น FALSE รูปแบบการใช้งาน ดังนี้

รูปแบบ

```
bool settype ( mixed &$var , string $type )
```

เมื่อ $var หมายถึง ตัวแปรที่ต้องการแปลงชนิด

$type หมายถึง ชนิดที่ต้องการแปลง มีรายละเอียดดังนี้

| | | |
|---|---|---|
| "boolean" | คือ | บูลีน |
| "integer" | คือ | เลขจำนวนเต็ม |
| "float" | คือ | เลขจำนวนจริง |
| "string" | คือ | ข้อความ |
| "array" | คือ | อาร์เรย์ |
| "object" | คือ | ออบเจ็กต์ |
| "null" | คือ | ค่าว่างหรือไม่กำหนดชนิดใดๆ |

**ตัวอย่างที่ 2.32** การใช้ฟังก์ชัน settype ( ) เพื่อแปลงหรือกำหนดชนิดข้อมูล

```
1   <?php
2       $foo = "5bar";              // เป็นชนิด string
3       $bar = true;                // เป็นชนิด boolean
4       settype ($foo, "integer");  // ผลลัพธ์ $foo จะมีค่าเท่ากับ 5 และเป็นชนิด integer
5       settype ($bar, "string");   // ผลลัพธ์ $bar จะมีค่าเท่ากับ "1" และเป็นชนิด string
6   ?>
```

จากตัวอย่างที่ 2.32 การใช้ฟังก์ชัน settype ( ) เพื่อแปลงหรือกำหนดชนิดข้อมูล อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $foo มีค่าเท่ากับ "5bar" โดยปริยายข้อมูลลักษณะเช่นนี้คือข้อมูลชนิดข้อความ

บรรทัดที่ 3 กำหนดให้ตัวแปร $bar มีค่าเท่ากับ true โดยปริยายข้อมูลลักษณะเช่นนี้คือข้อมูลชนิดตรรกะหรือบูลีน

บรรทัดที่ 4 ใช้ฟังก์ชัน settype เพื่อกำหนดให้ตัวแปร $foo แปลงเป็นข้อมูลชนิดเลขจำนวนเต็ม ฟังก์ชันจะพิจารณาว่าค่าภายในขึ้นต้นด้วยตัวเลขหรือไม่ เช่น จากตัวอย่างค่าภายในประกอบด้วยเลข 5 แล้วตามด้วยข้อความ ฟังก์ชันจะตัดเลือกเฉพาะตัวเลขแล้วลบ ข้อความทิ้งไป ดังนั้นค่าใหม่ที่กำหนดให้ตัวแปร $foo จึงเป็นชนิดเลขจำนวนเต็มและมีค่าเท่ากับ 5

บรรทัดที่ 5 ใช้ฟังก์ชัน settype เพื่อกำหนดให้ตัวแปร $bar เป็นชนิดข้อความ จากเดิมเป็นชนิดตรรกะหรือบูลีน ดังนั้นเมื่อเปลี่ยนแปลงชนิดใหม่ค่าภายในตัวแปรจะเปลี่ยนเป็น 1 หากก่อนหน้าค่าภายใน ตัวแปรเป็นชนิด false เมื่อปรับเปลี่ยนเป็นชนิดข้อความจะกลายเป็น 0

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

### 2.7.3 ฟังก์ชันตรวจสอบชนิดของข้อมูล (Type Identifier Functions)

ฟังก์ชันที่ใช้สำหรับตรวจสอบชนิดของข้อมูล ประกอบด้วย

**ตารางที่ 2.3** ฟังก์ชันที่ใช้สำหรับตรวจสอบชนิดของข้อมูล

| ชื่อฟังก์ชัน | หน้าที่ |
|---|---|
| is_array ( ) | ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดอาร์เรย์จริงหรือไม่ |
| is_bool ( ) | ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดตรรกะหรือบูลีนจริงหรือไม่ |
| is_float ( ) หรือ is_real | ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดเลขจำนวนจริงจริงหรือไม่ |
| is_integer ( ) หรือ is_int ( ) | ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดเลขจำนวนเต็มจริงหรือไม่ |
| is_null ( ) | ตรวจสอบชนิดของข้อมูลว่าเป็นค่าว่างจริงหรือไม่ |
| is_numeric ( ) | ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดตัวเลขจริงหรือไม่ |
| is_object ( ) | ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดวัตถุจริงหรือไม่ |
| is_resource ( ) | ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดติดต่อใช้ทรัพยากรของระบบจริงหรือไม่ |
| is_scalar ( ) | ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดข้อมูลพื้นฐานที่ภาษา PHP รองรับจริงหรือไม่ ชนิดข้อมูลพื้นฐานประกอบด้วย Boolean, integer, float และ string สำหรับ array, object, null และ resource ไม่ถือว่าเป็นชนิดข้อมูลพื้นฐาน |
| is_string ( ) | ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดข้อความจริงหรือไม่ |

จากตารางที่ 2.3 ฟังก์ชันที่ใช้สำหรับตรวจสอบชนิดของข้อมูล ทุกฟังก์ชันจะมีรูปแบบการใช้งานเหมือนกัน ประกอบด้วย ชื่อฟังก์ชันตรวจสอบ และในพารามิเตอร์ให้ใส่ตัวแปรที่ต้องการตรวจสอบ หลังจากที่ฟังก์ชันทำการตรวจสอบแล้ว และจะส่งค่ากลับเป็น true หากค่าของตัวแปรตรงกับชนิดข้อมูลที่ตรวจสอบ หากไม่ตรงกับชนิดข้อมูลที่ตรวจสอบจะส่งค่ากลับเป็น false การใช้ฟังก์ชันตรวจสอบชนิดข้อมูล นิยมใช้ร่วมกับโครงสร้างควบคุมเพื่อตรวจสอบเงื่อนไข เช่น โครงสร้างเงื่อนไข if เป็นต้น ดังนี้

รูปแบบ

```
boolean is_name (mixed var)  // เมื่อกำหนดให้ is_name แทนชื่อฟังก์ชันตรวจสอบ
```

เมื่อ $var หมายถึง ตัวแปรที่ต้องการตรวจสอบชนิด

**ตัวอย่างที่ 2.33** การใช้ฟังก์ชัน is_bool ( ) ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดตรรกะหรือบูลีน

```
1   <?php
2       $a = false;
3       if (is_bool($a) === true) {
4           echo "Yes, this is a boolean";
5       } else {
6           echo "No, this is a ",gettype($a);
7       }
8   ?>
```

ผลลัพธ์

```
Yes, this is a boolean
```

จากตัวอย่างที่ 2.33 การใช้ฟังก์ชัน is_bool ( ) อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $a มีค่าเท่ากับ false

บรรทัดที่ 3 ใช้ฟังก์ชัน is_bool ( ) ร่วมกับโครงสร้างเงื่อนไข if โดยฟังก์ชัน is_bool ( ) ใช้สำหรับตรวจสอบว่าตัวแปรหรือค่าของข้อมูลที่ระบุเป็นชนิดตรรกะหรือบูลีนจริงหรือไม่ หากผลของการตรวจสอบเป็นจริงตามเงื่อนไข จะไปทำงานต่อบรรทัดที่ 4 เมื่อทำตามคำสั่งโครงสร้างเงื่อนไข if เสร็จเรียบร้อย ทำงานต่อหลังบรรทัดที่ 7

บรรทัดที่ 4 แสดงคำว่า Yes, this is a boolean เมื่อทำงานตามคำสั่งเสร็จจะออกจากโครงสร้างเงื่อนไข if ทำงานต่อหลังบรรทัดที่ 7

บรรทัดที่ 5 ในกรณีที่เป็นเท็จจะเข้าสู่เงื่อนไขของโครงสร้างเงื่อนไข else

บรรทัดที่ 6 แสดงคำว่า No, this is a แล้วตามด้วยฟังก์ชัน gettype ($a) เพื่อแสดงชนิดของตัวแปร เมื่อทำงานเสร็จก็จะไปบรรทัดที่ 7 เพื่อออกจากโครงสร้างเงื่อนไข if...else

บรรทัดที่ 7 สิ้นสุดโครงสร้างเงื่อนไข if...else

บรรทัดที่ 8 สิ้นสุดสคริปต์ PHP

**ตัวอย่างที่ 2.34** การใช้ฟังก์ชัน is_int ( ) ตรวจสอบชนิดของข้อมูลว่าเป็นชนิดเลขจำนวนเต็ม

```
1    <?php
2        if (is_int (23)) {
3            echo "is an integer";
4        } else {
5            echo "is not an integer";
6        }
7    ?>
```

ผลลัพธ์

```
is an integer
```

จากตัวอย่างที่ 2.34 การใช้ฟังก์ชัน is_int ( ) อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 ใช้ฟังก์ชัน is_int( ) ร่วมกับโครงสร้างเงื่อนไข if โดยฟังก์ชัน is_int ใช้สำหรับตรวจสอบว่าตัวแปรหรือค่าของข้อมูลที่ระบุเป็นเลขจำนวนเต็มจริงหรือไม่ หากผลของการตรวจสอบเป็นจริงตามเงื่อนไข จะไปทำงานต่อบรรทัดที่ 3 หากเป็นเท็จจะไปทำงานต่อบรรทัดที่ 4

บรรทัดที่ 3 แสดงคำว่า is an integer (เป็นเลขจำนวนเต็ม) เมื่อทำงานตามคำสั่งเสร็จจะออกจากโครงสร้างเงื่อนไข if...else ทำงานต่อหลังบรรทัดที่ 6

บรรทัดที่ 4 ในกรณีที่เป็นเท็จจะเข้าสู่โครงสร้างเงื่อนไข else

บรรทัดที่ 5 แสดงผลคำว่า is not an integer (ไม่ใช่เลขจำนวนเต็ม)

บรรทัดที่ 6 สิ้นสุดโครงสร้างเงื่อนไข if...else

บรรทัดที่ 7 สิ้นสุดสคริปต์ PHP

**ตัวอย่างที่ 2.35** ตัวอย่างการใช้ฟังก์ชัน is_ resource ( )

```
1    <?php
2        $db_link = mysql_connect("localhost","mysql_user","mysql_pass");
3        if (!is_resource($db_link)) {
4            die("Can\'t connect" . mysql_error());
5        }
6    ?>
```

จากตัวอย่างที่ 2.35 การใช้ฟังก์ชัน is_ resource ( ) เพื่อตรวจสอบการเชื่อมต่อกับฐานข้อมูล MySQL ว่าสามารถเชื่อมต่อได้หรือไม่ อธิบายดังนี้

บรรทัดที่ 1 เริ่มต้นสคริปต์ PHP

บรรทัดที่ 2 กำหนดให้ตัวแปร $db_link เป็นชนิด resource เพื่อเชื่อมต่อกับฐานข้อมูล MySQL

บรรทัดที่ 3 ตรวจสอบเงื่อนไข โดยใช้โครงสร้างเงื่อนไข if ร่วมกับฟังก์ชัน is_resource ($db_link) หากไม่สามารถเชื่อมต่อได้ แสดงว่า ตัวแปร $db_link ปิดการเชื่อมต่อกับฐานข้อมูล อาจมีสาเหตุมาจากไม่สามารถเชื่อมกับฐานข้อมูล MySQL อันเนื่องมาจากชื่อโฮส หรือชื่อผู้ใช้งาน หรือรหัสผ่านสำหรับการเชื่อมต่อฐานข้อมูล MySQL ไม่ถูกต้อง ดังนั้นเมื่อใช้ฟังก์ชัน is_resource เพื่อตรวจสอบตัวแปร $db_link หากส่งผลค่ากลับเป็น false แสดงว่าเชื่อมต่อไม่ได้ หากเป็น true แสดงว่าเชื่อมต่อได้ แต่เมื่อใช้งานร่วมกับโครงสร้างเงื่อนไข if เงื่อนไขจะต้องเป็นจริงเท่านั้นถึงจะเข้าไปทำคำสั่งภายในเงื่อนไข จึงเป็นเหตุผลที่ต้องการใช้เครื่องหมาย ! (not) เพื่อทำเป็นเงื่อนไขปฏิเสธหรือตรงกันข้าม เช่น !true คือ false และ !false คือ true โดยสรุป หากเงื่อนไขของการตรวจสอบด้วยโครงสร้างเงื่อนไข if ได้ผลเป็น !false จะทำให้เงื่อนไขเป็นจริง จะไปทำงานต่อบรรทัดที่ 4 หากเป็นเท็จจากเงื่อนไข คือ สามารถเชื่อมต่อฐานข้อมูลได้จะออกจากโครงสร้างเงื่อนไข if คือ ทำงานต่อหลังจากบรรทัดที่ 5 (หากมีคำสั่งอื่นๆ เพิ่มเติม)

บรรทัดที่ 4 สั่งให้โปรแกรมหยุดทำงาน โดยใช้ฟังก์ชัน die( ) แล้วแจ้งข้อความสาเหตุที่ไม่สามารถเชื่อมต่อฐานข้อมูลได้

บรรทัดที่ 5 สิ้นสุดโครงสร้างเงื่อนไข if

บรรทัดที่ 6 สิ้นสุดสคริปต์ PHP

## 2.8 คำสงวนของภาษา PHP

คำสงวนของภาษา PHP (Reserved Words in PHP) หมายถึง คำที่จะใช้เป็นคำสั่งเฉพาะของภาษา PHP ดังนั้นต้องไม่นำคำเหล่านี้ไปตั้งหรือกำหนดเป็นชื่อให้กับตัวแปร หรือฟังก์ชัน มิฉะนั้นจะทำให้เกิดข้อผิดพลาดขึ้นได้มีรายละเอียด ดังตารางที่ 2.4

**ตารางที่ 2.4** คำสงวนของภาษา PHP

| | | | | |
|---|---|---|---|---|
| and | or | xor | _FILE_ | exception (5) |
| _LINE_ | array ( ) | as | break | case |
| class | const | continue | declare | default |
| die ( ) | do | echo | else | elseif |
| empty ( ) | enddeclare | endfor | endforeach | endif |
| endswitch | endwhile | eval ( ) | exit ( ) | extends |

**ตารางที่ 2.4** (ต่อ)

| For | foreach | function | global | if |
|---|---|---|---|---|
| include ( ) | include_once ( ) | isset ( ) | list ( ) | new |
| print | require ( ) | require_once ( ) | return ( ) | static |
| switch | unset ( ) | use | var | while |
| _FUNCTION_ | _CLASS_ | _METHOD_ | nal ( ) | php_user_lter (5) |
| interface (5) | implements (5) | extends | public (5) | private (5) |
| protected (5) | abstract (5) | clone (5) | try (5) | catch (5) |
| throw (5) | cfunction (4 only) | old_function (4 only) | | |

**สรุป**

เครื่องมือที่ใช้เขียนสคริปต์ PHP สามารถเลือกใช้ได้หลายโปรแกรม เช่น Notepad, Notepad++, EditPlus, vi และ Adobe Dreamweaver เป็นต้น รูปแบบการเขียนสคริปต์ PHP นั้น จะใช้เทคนิควิธีการแทรกประโยคคำสั่งร่วมกับ HTML tags จุดที่สำคัญ คือ จุดเริ่มต้นและจุดสิ้นสุดของสคริปต์ PHP ซึ่งหลักๆ การบอกจุดเริ่มต้นและจุดสิ้นสุดของสคริปต์ PHP มี 4 รูปแบบ ประกอบด้วย 1) XML Style หรือ Default Syntax 2) SGML Style หรือ Short Tags 3) Script Style และ 4) ASP Style และประโยคคำสั่งที่เขียนในแต่ละบรรทัดเรียกว่าสคริปต์คำสั่ง แต่ละสคริปต์คำสั่งจะต้องใส่เครื่องหมาย ; (Semicolon) เพื่อบอกจุดสิ้นสุดในแต่ละสคริปต์คำสั่ง ชนิดของข้อมูลที่รองรับในภาษา PHP มีหลายชนิด ดังนั้นก่อนการเขียนสคริปต์ PHP ควรศึกษา ทำความเข้าใจ และเลือกใช้ชนิดของข้อมูลให้เหมาะสมตามลักษณะงาน เพื่อความถูกต้องในการประมวลผลและแสดงผลลัพธ์ตามต้องการ

**คำถามท้ายบท**

1. จงอธิบายเหตุผล และรูปแบบการเขียนสคริปต์ PHP แทรกใน HTML tags มีรูปแบบอะไรบ้าง พร้อมยกตัวอย่างประกอบในแต่ละรูปแบบ

2. จงอธิบายคำสั่งที่ใช้แสดงผลข้อมูลบนเว็บบราวเซอร์ที่สำคัญของภาษา PHP รูปแบบการใช้งาน และยกตัวอย่างประกอบ

3. จงอธิบายถึงตัวกำหนดชนิดการแสดงผลที่ใช้งานร่วมกับคำสั่ง printf พร้อมยกตัวอย่างประกอบ

4. จงอธิบายถึงชนิดข้อมูลที่รองรับในภาษา PHP รายละเอียดของแต่ละชนิดพอสังเขป พร้อมยกตัวอย่างประกอบ

5. จงอธิบายถึงฟังก์ชันเรียกดูชนิดของข้อมูล ฟังก์ชันแปลงชนิดข้อมูล และฟังก์ชันตรวจสอบชนิดของข้อมูล พร้อมยกตัวอย่างประกอบ