Chapter 2: Type Functions, Operators, Control Structures, and Loops

Presented by: Salima HASSAINE, Ph.D.

Part 1: Type Functions

- isset(…) Function
- unset(…) Function
- empty(…) Function
- gettype(…) Function
- settype(…) Function

NULL Value

NULL is used to represent the concept of nothing or the state of being empty. If a variable is created without a value, it is automatically assigned a value of NULL.

PHP has three functions to check or set to NULL value:

- **isset(…)**
- **unset(…)**
- **empty(…)**

isset(…) Function

Function isset($var) is used to test whether a **variable** has been **set** (initialized). It returns true or false.

```
<?php
        $var1 = 3;
        $var2 = "cat";
        $var3 = NULL;
        echo "var1 is set:". isset($var1) . "<br />" ;
        echo "var2 is set:". isset($var2) . "<br />" ;
        echo "var3 is set:". isset($var3) . "<br />" ;
?>
```

unset(…) Function

Function unset($var) is used to empty a variable by setting the value to NULL

```
<?php
        $var1 = 3;
        $var2 = "cat";
        $var3 = NULL;

        unset($var1) ;

        echo "var1 is set:". isset($var1) . "<br />" ;
        echo "var2 is set:". isset($var2) . "<br />" ;
        echo "var3 is set:". isset($var3) . "<br />" ;
?>
```

empty(…) Function

Function empty($var) is used to check if a variable is empty (NULL, initialized by zero, etc).

```
<?php
        $var1 = 0;
        $var2 = " ";
        $var3 = NULL;

        echo "var1 is set:". isset($var1) . "<br />" ;
        echo "var2 is set:". isset($var2) . "<br />" ;
        echo "var3 is set:". isset($var3) . "<br />" ;

        echo "var1 is set:". empty($var1) . "<br />" ;
        echo "var2 is set:". empty($var2) . "<br />" ;
        echo "var3 is set:". empty($var3) . "<br />" ;
?>
```

Type Casting: gettype(…)

Function gettype($var) will retrieve an item's type

```
<?php
        $var1 = 0;
        $var2 = "3";
        $var3 = NULL;

        echo "var1 type :". gettype($var1) . "<br />" ;
        echo "var2 type :". gettype($var2) . "<br />" ;
        echo "var3 type :". gettype($var3) . "<br />" ;

        $var4 = (int) $var2 ;

        echo "var4 type :". gettype($var4) . "<br />" ;
?>
```

Type Casting: settype(…)

Function settype($var) is used to change the type of a variable.

```
<?php
        $var1 = "1";
        $var2 =  2;
        echo "var1 type :". gettype($var1) . "<br />" ;
        echo "var2 type :". gettype($var2) . "<br />" ;

        settype($var1 , "int");
        settype($var2 , "string");

        echo "var1 type :". gettype($var1) . "<br />" ;
        echo "var2 type :". gettype($var2) . "<br />" ;
?>
```

Part 2: PHP Operators

- PHP Arithmetic Operators
- PHP Assignment Operators
- PHP Comparison Operators
- PHP Increment / Decrement Operators
- PHP Logical Operators
- PHP String Operators
- PHP Array Operators

PHP Arithmetic Operators

The PHP arithmetic operators **are used with numeric values** to **perform common arithmetical operations**.

Operator	**Name**	**Result**
$x + $y	Addition	Sum of $x and $y
$x - $y	Subtraction	Difference of $x and $y
$x * $y	Multiplication	Product of $x and $y
$x / $y	Division	Quotient of $x and $y
$x % $y	Modulus	Remainder of $x divided by $y
$x ** $y	Exponentiation	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

PHP Assignment Operators

The PHP assignment operators **are used with numeric values to write a value to a variable.**

Assignment	Same as...	Description
$x = $y	$x = $y	The left operand gets set to the value of the expression on the right
$x += $y	$x = $x + $y	Addition
$x -= $y	$x = $x - $y	Subtraction
$x *= $y	$x = $x * $y	Multiplication
$x /= $y	$x = $x / $y	Division
$x %= $y	$x = $x % $y	Modulus

PHP Comparison Operators

They are used to compare two values (number or string):

Operator	**Name**	**Result**
$x == $y	Equal	Returns true if $x is equal to $y
$x === $y	Identical	Returns true if $x is equal to $y, and they are of the same type
$x != $y	Not equal	Returns true if $x is not equal to $y
$x <> $y	Not equal	Returns true if $x is not equal to $y
$x !== $y	Not identical	Returns true if $x is not equal to $y, or they are not of the same type
$x > $y	Greater than	Returns true if $x is greater than $y
$x < $y	Less than	Returns true if $x is less than $y
$x >= $y	Greater than or equal	Returns true if $x is greater than or equal to $y
$x <= $y	Less than or equal to	Returns true if $x is less than or equal to $y

PHP Increment / Decrement Operators

These operators are used to increment or decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	**Name**	**Result**
$x and $y	And	True if both $x and $y are true
$x or $y	Or	True if either $x or $y is true
$x xor $y	Xor	True if either $x or $y is true, but not both
$x && $y	And	True if both $x and $y are true
$x || $y	Or	True if either $x or $y is true
! $x	Not	True if $x is not true

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Result
$x . $y	Concatenation	Concatenation of $x and $y
$x .= $y	Concatenation assignment	Appends $x to $y

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Result
$x + $y	Union	Union of $x and $y
$x == $y	Equality	Returns true if $x and $y have the same key/value pairs
$x === $y	Identity	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
$x != $y	Inequality	Returns true if $x is not equal to $y
$x <> $y	Inequality	Returns true if $x is not equal to $y
$x !== $y	Non-identity	Returns true if $x is not identical to $y

Part 3: Control Structures

- The if Statement
- The if...else Statement
- The if...elseif....else Statement
- The switch Statement

The if Statement

Syntax

```
if (condition) {
    code to be executed if condition is true;
}
```

```
<?php
        date_default_timezone_set('America/Montreal');
        $v_date = date('l jS \of F Y H:i:s A');
        echo "Current Date and Time is : $v_date <br/>";

        $time = date("H");

        if ($time < "20") {
                echo "Have a good day!";
        }
?>
```

The if...else Statement

Syntax

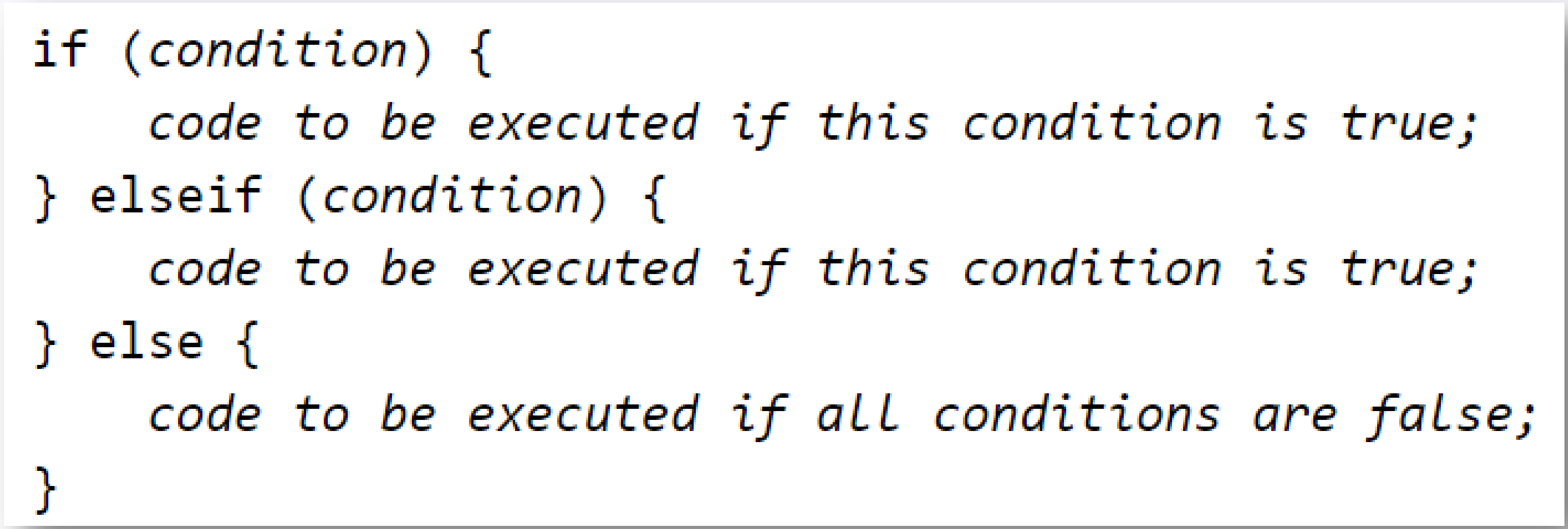
```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

```php
<?php
        $time = date("H");

        if ($time < "20") {
                echo "Have a good day!";
        } else {
                echo "Have a good night!";
        }
?>
```

The if...elseif....else Statement

Syntax



```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

Example

```php
<?php
        $time = date("H");

        if ($time < "10") {
                echo "Have a good morning!";
        } elseif ($time < "20") {
                echo "Have a good day!";
        } else {
                echo "Have a good night!";
        }
?>
```

The switch Statement

Syntax

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

Example

```
<?php
        $size= “S";
        switch ($size) {
        case "S":
                echo "Your size is small !";
                break;
        case "M":
                echo "Your size is medium !";
                break;
        case "L":
                echo "Your size is large !";
                break;
        default:
                echo "Your size is undefined!";
}
?>
```

Part 4: Loops

- While Loop
- do...while Loop
- for Loop
- foreach Loop

While Loop

Syntax

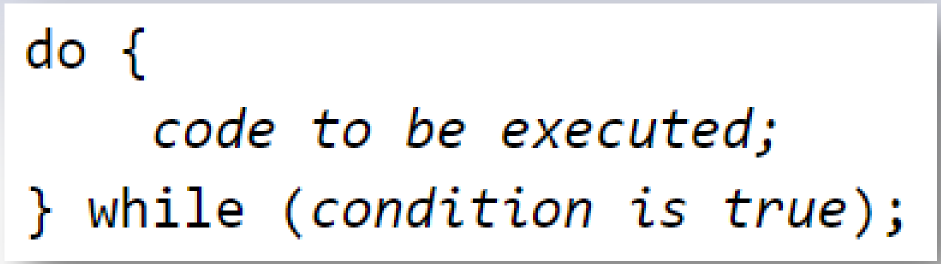
```
while (condition is true) {
    code to be executed;
}
```

```
<?php
        $x = 1;

        while ($x <= 5) {
                echo "The number is: $x <br />";
                $x ++;
        }
?>
```

do ...While Loop

Syntax



```
do {
    code to be executed;
} while (condition is true);
```

```
<?php
        $x = 1;

         do {
                echo "The number is: $x <br />";
                $x ++;
        } while ($x <= 5) ;
?>
```

PHP for Loop

Syntax

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
```

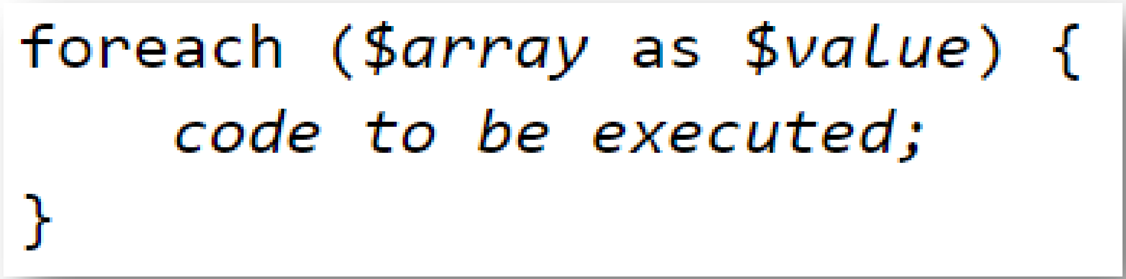
- ***init counter***: Initialize the loop counter value
- ***test counter***: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- ***increment counter***: Increases the loop counter value

```
<?php
        for ($x = 0; $x <= 10; $x++) {
                echo "The number is: $x <br />";
        }
?>
```

PHP foreach Loop

The foreach loop works only on arrays, and is used to Loop through each key/value pair in an array

Syntax



```
foreach ($array as $value) {
    code to be executed;
}
```

For every loop iteration, the value of the **current array element** is assigned to **$value** and the array pointer is moved by one, until it reaches the last array element.

Example

```
<?php
        $colors = array("red", "green", "blue", "yellow");

        foreach ($colors as $value) {
                echo "$value <br />";
        }
?>
```

Lab

Questions?