Control Structure

H S Rana

CIT,UPES

January 31, 2014

Data types in PHP

- PHP supports eight primitive data types
- There are four scalar types
  - boolean
  - integer
  - floating-point number
  - string
- There are two structured types
  - array
  - object
- There are two special data types
  - resource
  - NULL
- The programmer does not specify the type of a variable
- a variable's type is determined from the context of its usage

Setting and Checking Variable Data Types

PHP automatically determines a variable's data type from the content it holds. And if the variable's content changes over the duration of a script, the language will automatically set the variable to the appropriate new data type.

```
<?php
// define string variable
$whoami = 'Sarah';
echo gettype($whoami); // output: 'string'
// assign new integer value to variable
$whoami = 99.8;
echo gettype($whoami);  // output: 'double'
unset($whoami);  // destroy variable
echo gettype($whoami); // output: 'NULL'
?>
```

Type Casting

it's possible to explicitly set the type of a PHP variable by casting a variable to a specific type before using it.

```
<?php
// define floating-point variable
$speed = 501.789;

// cast to integer
$newSpeed = (integer) $speed;

// output: 501
echo $newSpeed;
?>
```

Checking Data Type

In addition to the gettype() function, PHP includes a number of more specialized functions, to test if a variable is of a specific type.

Function	Purpose
`is_bool()`	Tests if a variable holds a Boolean value
`is_numeric()`	Tests if a variable holds a numeric value
`is_int()`	Tests if a variable holds an integer
`is_float()`	Tests if a variable holds a floating-point value
`is_string()`	Tests if a variable holds a string value
`is_null()`	Tests if a variable holds a NULL value
`is_array()`	Tests if a variable is an array
`is_object()`	Tests if a variable is an object

Using Constant

Constants are defined using PHP's *define()* function, which accepts two arguments: the name of the constant, and its value. Constant names must follow the same rules as variable names, with one exception: the $ prefix is not required for constant names.

Using Constant

Constants are defined using PHP's *define()* function, which accepts two arguments: the name of the constant, and its value. Constant names must follow the same rules as variable names, with one exception: the $ prefix is not required for constant names.

```
<?php
// define constants
define ('PROGRAM', 'The Matrix');
define ('VERSION', 11.7);

// use constants
// output: 'Welcome to The Matrix (version 11.7)'
echo 'Welcome to ' . PROGRAM . ' (version ' . VERSION . ')';
?>
```

Expressions

- Expressions evaluate to a value. The value can be a string, number, boolean, etc...
- Expressions often use operations and function calls, and there is an order of evaluation when there is more than one operator in an expression
- Expressions can also produce objects like arrays

Operators

High

Operator(s)	Type
( )	Parentheses
++ --	Increment/Decrement
!	Logical
* / %	Arithmetic
+ - .	Arithmetic and String
<< >>	Bitwise
< <= > >= <>	Comparison
== != === !==	Comparison
&	Bitwise (and references)
^	Bitwise
|	Bitwise

Operator Precedence

&&	Logical
||	Logical
? :	Ternary
= += -= *= /= .= %= &= != ^= <<= >>=	Assignment
and	Logical
xor	Logical
or	Logical

Low

String concatenation

String concatenation uses the period (.) to append one string of characters to another.

$echo\ \text{“}hello\text{”}.\text{”}world\text{”};$

$\$x=\text{”}hello\text{”};$

$\$x.=\text{”}world\text{”}$

The if Statement

The simplest of PHP's conditional statements is the if statement. This works much like the English-language statement, "if X happens, do Y."

```
<?php
// if number is less than zero
// print message
$number = -88;
if ($number < 0) {
    echo 'That number is negative';
}
?>
```

The if-else Statement

```
<?php
// change message depending on whether
// number is less than zero or not
$number = -88;
if ($number < 0) {
    echo 'That number is negative';
} else {
    echo 'That number is either positive or zero';
}
?>
```

compact way to write an if-else statement

The Standard if-else Block	The Equivalent Block Using the Ternary Operator

<?php
if ($x < 10) {
echo 'X is less than 10';
} else {
echo 'X is more than 10';
}
?>
	<?php
	echo ($x < 10) ? 'X is less than 10' :
	'X is more than 10';
	?>

The if-elseif-else Statement

To test the multiple condition we use if-elseif-else Statement.

```
<?php
$today = 'Tuesday';
if ($today == 'Monday') {
    echo 'Monday\'s child is fair of face.';
} elseif ($today == 'Tuesday') {
    echo 'Tuesday\'s child is full of grace.';
} elseif ($today == 'Wednesday') {
    echo 'Wednesday\'s child is full of woe.';
} elseif ($today == 'Thursday') {
    echo 'Thursday\'s child has far to go.';
} elseif ($today == 'Friday') {
    echo 'Friday\'s child is loving and giving.';
} elseif ($today == 'Saturday') {
    echo 'Saturday\'s child works hard for a living.';
} else {  echo 'No information available for that day';
}
```

The switch-case Statement

An alternative to the if-elseif-else statement is the switch-case statement, which does almost the same thing: it tests a variable against a series of values until it finds a match, and then executes the code corresponding to that match

The switch-case Statement

An alternative to the if-elseif-else statement is the switch-case statement, which does almost the same thing: it tests a variable against a series of values until it finds a match, and then executes the code corresponding to that match

```
<?php

$today = 'Tuesday';
switch ($today) {
    case 'Monday':
        echo 'Monday\'s child is fair of face.';
        break;
    case 'Tuesday':
        echo 'Tuesday\'s child is full of grace.';
        break;
```

The switch-case Statement

```
{case 'Wednesday':
        echo 'Wednesday\'s child is full of woe.';
        break;
    case 'Thursday':
        echo 'Thursday\'s child has far to go.';
        break;
    case 'Friday':
        echo 'Friday\'s child is loving and giving.';
        break;
    case 'Saturday':
        echo 'Saturday\'s child works hard for a living.';
        break;
    default:
        echo 'No information available for that day.';
        break;
}
?>
```

Combining Conditional Statements

- PHP allows one conditional statement to be nested within another
- You can also combine conditional statements by using logical operators, such as the && or || operator.
-
```
<?php
// for employees with annual comp <= $15000
// those with a rating >= 3 get a $5000 bonus
// everyone else gets a $3000 bonus
if ($rating >= 3) {
    if ($salary < 15000) {
        $bonus = 5000;
    }
} else {
    if ($salary < 15000) {
        $bonus = 3000;
    }
}
?>
```

Combining Conditional Statements

```
<?php
$year = 2008;
// leap years are divisible by 400
// or by 4 but not 100
if (($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 ==
    echo "$year is a leap year.";
} else {
    echo "$year is not a leap year.";
}
?>
```

Loops in PHP

PHP has following loops

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

while Loop

```
<?php
// repeat continuously until counter becomes 10
// output: 'xxxxxxxxx'
$counter = 1;
while ($counter < 10) {
  echo 'x';
  $counter++;
}
?>
```

do-while loop

```
<?php
// repeat continuously until counter becomes 10
// output: 'xxxxxxxxx'
$counter = 1;
do {
  echo 'x';
  $counter++;
} while ($counter < 10);
?>
```

for loop

```
<?php
// repeat continuously until counter becomes 10
// output:
for ($x=1; $x<10; $x++) {
  echo "$x ";
}
?>
```