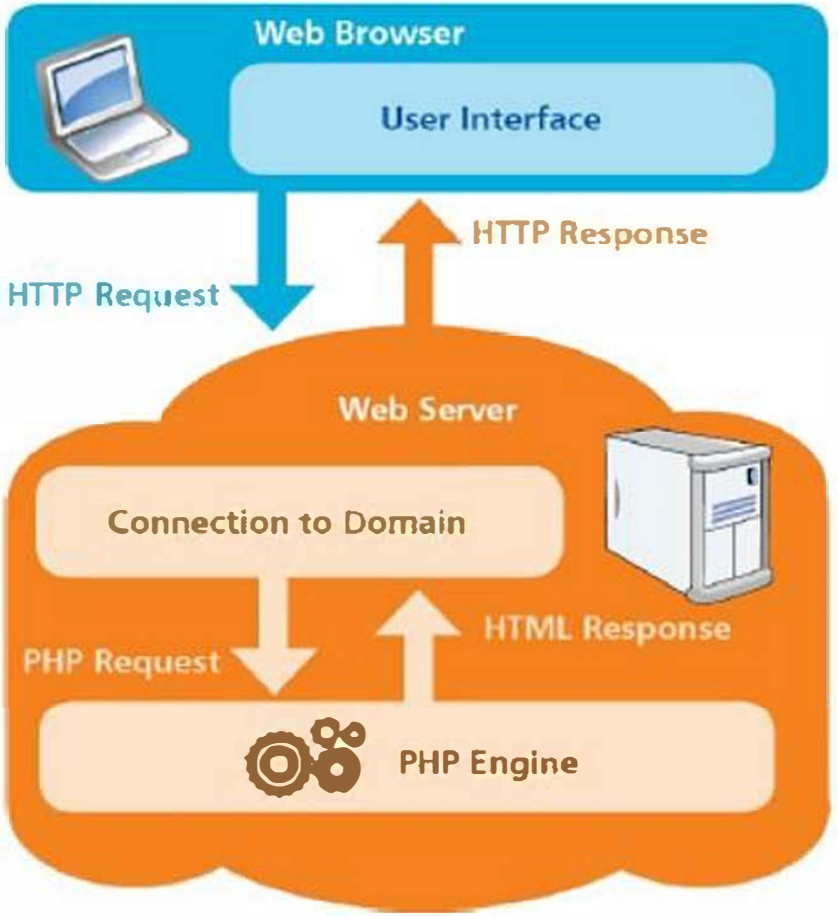
Introduction to PHP

Bernard G. Sanidad, DBA, MIT

Learning Objectives

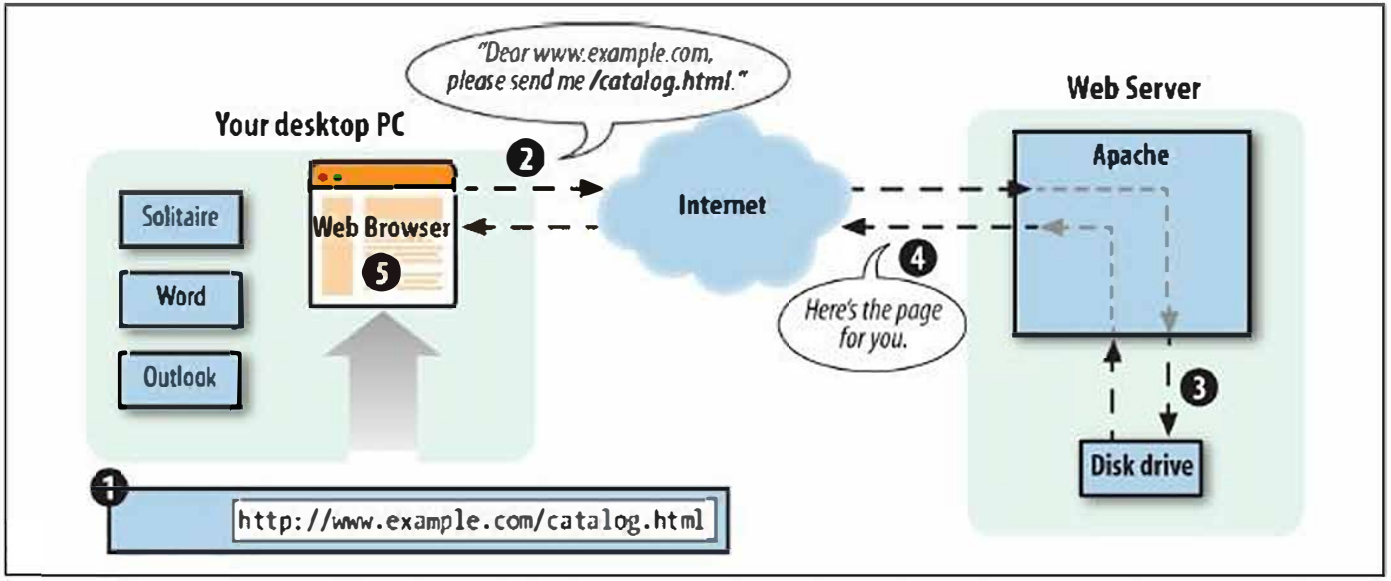
- At the end of the discussion, the students would be able to:
  - Discuss how php fits in the interaction between web browser and server;
  - Compare the difference of PHP to other scripting language;
  - Identify the building blocks of PHP;
  - Discuss the PHP language basics: syntax, variables, and operators.
  - Create a PHP script base on the lessons learned.

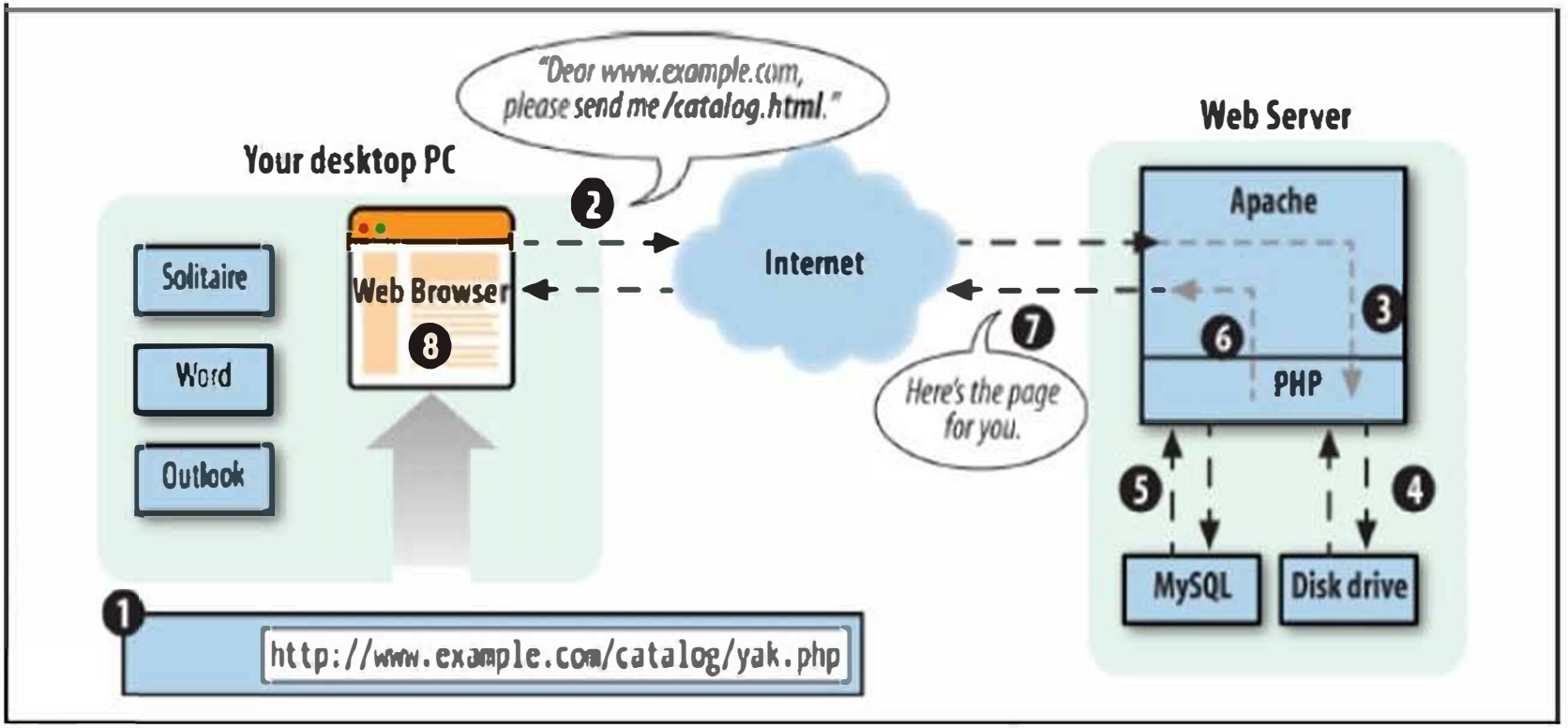
Understanding the Cloud



Bernard G. Sanidad, DBA, MIT

How PHP fits into the interaction between a web browser and web server





Bernard G. Sanidad, DBA, MIT

Client/Server on the World Wide Web

- Standard web sites operate on a request/response basis.
- A user requests a resource E.g. HTML document
- Server responds by delivering the document to the client
- The client processes the document and displays it to user

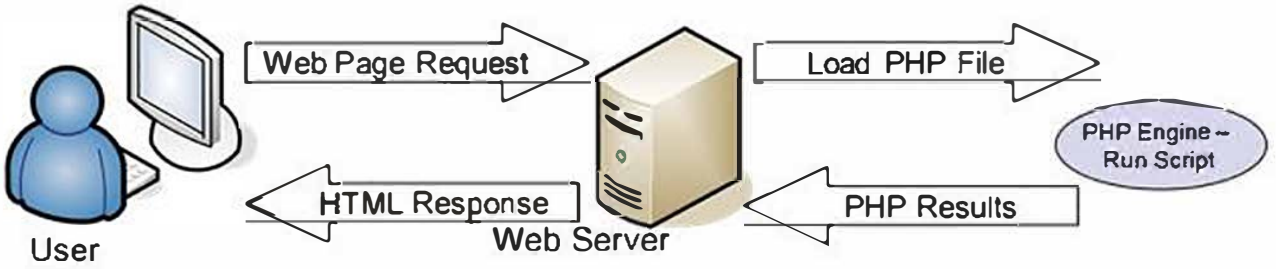
Server Side Programming

- Provides web site developers to utilise resources on the web server
- Non-public resources do not require direct access from the clients
- Allows web sites to be client agnostic (unless JavaScript is used also)
- Most server side programming script is embedded within markup (although does not have to be, sometimes better not to)

PHP

- PHP: **P**HP **H**ypertext **P**re-processor
- Programming language that is interpreted and executed on the server
- Execution is done before delivering content to the client
- Contains a **vast** library of functionality that programmers can harness
- Executes entirely on the server, requiring no specific features from the client

- Static resources such as regular HTML are simply output to the client from the server
- Dynamic resources such as PHP scripts are processed on the server prior to being output to the client
- PHP has the capability of connecting to many database systems making the entire process transparent to the client



Common Uses of PHP

- PHP performs system functions, *i.e.* from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, *i.e.* gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Why Use PHP?

- PHP is free
- PHP is Cross-flatform
- PHP is widely used
- PHP hides its complexity
- PHP is built for web programming

Characteristics of PHP

1. Simplicity
2. Efficiency
3. Security
4. Flexibility
5. Familiarity

Syntax and Structure

- Syntax:

```
<?php
        statements;
?>
```

```php
<?php
        echo "Good Afternoon";
        echo "Thomasian";
        echo "!";
?>
```

Structure

- PHP is similar to C
- All scripts start with <?php and with with ?>
- Line separator: ; (semi-colon)
- Code block: { //code here } (brace brackets)
- White space is generally ignored (not in strings)
- Comments are created using:
  - // single line quote
  - /* Multiple line block quote */
- Precedence
  - Enforced using parentheses
  - E.g. $sum = 5 + 3 * 6; // would equal 23
  - $sum = (5 + 3) * 6; // would equal 48

Declaring and Initializing Variables

- Prefixed with a $
- Assign values with = operator
  - $var_name = value;
  - Example:
    - $name = “Johnny”;
    - $amountDue = 15;
- No need to define type
- Variable names are case sensitive
  - $author and $Author are different

```
<?php
        $votingAge = 18;

        echo $votingAge;
```

Naming Rules for Variables

- A variable name must start with a letter or an underscore "_“
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string), or with capitalization ($myString)
- There is no size limit for variables.

String Variables

- String variables are used for values that contains characters.
- Example:

```
<?php
    $message = “Good Morning!”;
    echo $message;
?>
```

```
<!DOCTYPE html>
<html>
        <head>
                <title>Integration of PHP to HTML </title>
        </head>

        <body>
                <?php
                        echo "Good Day everyone!";
                ?>
        </body>
</html>
```

Superglobal Variables

- **$_ENV[]** - This is an array that contains environment variables.
- **$_GET[]** - This array holds all of the "GET" variables gathered from the user's web
- browser.
- **$_POST[]** - This superglobal is similar to $_GET. The only difference is that $_POST involves POST variables only.
- **$_SERVER** – This kind of array holds the values of web-server variables.

Defining Constants

- Constant – contains information that does not change during the course of program execution.
  - Use define() function
  - Should not include a dollar sign.
- Syntax:

```
define ("CONSTANT_NAME", value);
```

- Example:
  - define (“DEFAULT_LANGUAGE”, “English”);
  - define (“VOTING_AGE”, 18);
  - define (“DEFAULT_LANGUAGE”, “English”, “TRUE”);

```
<!DOCTYPE html>
<html>
        <head>
                <title> PHP CODE BLOCKS </title>
        </head>
        <body>
                <?php
                        $worldVar = "World";
                        $sunVar = "Sun";
                        $moonVar = "Moon";
                        $worldInfo = 92897000;
                        $sunInfo = 72000000;
                        $moonInfo = 3456;

                        echo "<p> Hello $worldVar! <br/>";
                        echo "The $worldVar is $worldInfo miles from the $sunVar. <br/>";

                        echo "Hello ", $sunVar, "! <br/>";
                        echo "The $sunVar's core temperature is approximately $sunInfo degrees fahrenheit. <br/> ";

                        echo "Hello ", $moonVar, "! <br/>";
                        echo "The $moonVar is $moonInfo miles in diameter. </p>";
                ?>
                </body>
</html>
```

Adding Comments to PHP

1. **Line comment** – automatically terminates at the end of two forward slashes (//) or the pound symbol (#) before the text you want to use as a comment.

e.g.

//This line comment takes up an entire line

#This is another way of creating a line comment

2. **Block comments** – allow multiple lines of comment text to be added. (/* */

/* This is another way of creating a block comment */

Concatenation Operator

```
<?php
    $message1 = “Mabuhay!”;
    $message2 = “Welcome to the Philippines!”
    echo $message1 . “ “ . $message2;
?>
```

```
<!DOCTYPE html>
<html>
        <head>
                <title> PHP CODE BLOCKS </title>
        </head>
        <body>
                <?php
                        $worldVar = "World";
                        $sunVar = "Sun";
                        $moonVar = "Moon";
                        define("WORLD_INFO", 92897000);
                        $sunInfo = 72000000;
                        $moonInfo = 3456;

                        echo "<p> Hello $worldVar! <br/>";
                        //echo "The $worldVar is $worldInfo miles from the $sunVar. <br/>";

                        echo "The $worldVar is ", WORLD_INFO, " miles from the $sunVar. <br/>";

                        echo "Hello ", $sunVar, "! <br/>";
                        echo "The $sunVar's core temperature is approximately $sunInfo degrees fahrenheit. <br/> ";

                        echo "Hello ", $moonVar, "! <br/>";
                        echo "The $moonVar is $moonInfo miles in diameter. </p>";
                ?>
                </body>
</html>
```

- Acknowledgement:
- www.freepik.com
- www.codeauthority.com
- www.vectorsctock.com

Microsoft Visual Studio

PHP: Data Types & Operators

Bernard G. Sanidad, DBA, MIT

Learning Objectives

- At the end of the discussion, the students would be able to:
  - Enumerate the common data types in PHP;
  - Differentiate the types of operators use in PHP;
  - Demonstrate how data types and operators are use in the creation of script;
  - Apply the lessons learned in the development of a web page.

Something to Ponder

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.

Example

```
<?php
        $body_temperaure = 98.6;
        echo $body_temperaure;
        echo “<p> The body temperature  is
$body_temperature degrees Fahrenheit”;
        $body_temperature = 37.0;
//Statement assigning a value
        Echo “($body_temperature degrees
Celsius) </p>”;
        ?>
```

PHP Data Types

- **Integers** – are whole numbers, without a decimal point, like 4195.
- **Doubles** – are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** – have only two possible values either true or false.
- **NULL** – is a special type that only has one value: NULL.
- **Strings** – are sequences of characters, like 'PHP supports string
- operations.'

Double

- <?php
- $many = 2.2888800;
- $many_2 = 2.2111200;
- $few = $many + $many_2;
- print("$many + $many_2 = $few
");
- ?>

String

- Similar in form and behavior to Perl and the Unix shell, strings can be initialized in three ways:
  - with single quotes
  - with double quotes
  - “here document” (heredoc) format.
- With single-quoted strings, the only special characters you need to escape inside a string are backslash and the single quote itself.

String

- The escape-sequence replacements are −
  - \n is replaced by the newline character
  - \r is replaced by the carriage-return character
  - \t is replaced by the tab character
  - \$ is replaced by the dollar sign itself ($)
  - \" is replaced by a single double-quote (")
  - \\ is replaced by a single backslash (\)
  - \0 add the 0 character of ASCII
  - \x hexadecimal representation of the character

- print "I've gone to the store.";
- print "The sauce cost \$10.25.";
- $cost = '$10.25';
- print "The sauce cost $cost.";
- print "The sauce cost \$\061\060.\x32\x35.";

- If the value is a number, it is false if exactly equal to zero and true
- otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
- Don't use double as Booleans.

- **Arrays** – are named and indexed collections of other values.
- **Objects** – are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** – are special variables that hold references to resources external to PHP (such as database connections).

Building Expression

- **Expression** – is a literal value or variable (or a combination of literal values, variables, operators, and other expressions) that can be evaluated by the PHP scripting engine to produce a result.
- **Operands** – are variables and literals contained in an expression.
- **Literal** – static value such as a string or a number.
- **Operator** – are symbols such as addition operator (+), and multiplication operator (*), which are used in expression to manipulate operands.

PHP Operators

Type	Description
Array	Performs operations on arrays
Arithmetic	Performs mathematical calculations
Assignment	Assigns values to variables
Comparison	Compares operands and returns a Boolean value
Logical	Performs Boolean operations on Boolean operands
Special	Performs various tasks; these operators do not fit within other operator categories
String	Performs operations on strings

Arithmetic Operators

Symbol	Operation	Description
+	Addition	Adds to operands
-	Subtraction	Subtracts the right operand from the left operand
*	Multiplication	Multiplies two operands
/	Division	Divides the left operand by the right operand
%	Modulus	Divides the left operand by the right operand and returns the remainder

Arithmetic Unary Operator

Symbol	Operation	Description
++	Increment	Increases an operand by a value of 1
--	Decrement	Decreases an operand by a value of 1

Assignment Operators

Symbol	Operation	Description
=	Assignment	Assigns the value of the right operand to the left operand
+=	Compound addition assignment	Adds the value of the right operand to the value of the left operand and assigns the new value to the left operand
-=	Compound subtraction assignment	Subtracts the value of the right operand from the value of the left operand and assigns the new value to the left operand
*=	Compound multiplication assignment	Multiplies the value of the right operand by the value of the left operand and assigns the new value to the left operand
/=	Compound division assignment	Divides the value of the left operand by the value of the right operand and assigns the new value to the left operand
%=	Compound modulus assignment	Divides the value of the left operand to the value of the right operand and assigns the remainder (modulus) to the left operand

Comparison Operators

Symbol	Operation	Description
==	Equal	Returns TRUE if the operands are equal
===	Strict equal	Returns TRUE if the operands are equal of the same data type
!= or <>	Not equal	Returns TRUE if the operands are not equal
!==	Strict not equal	Returns TRUE if the operands are not equal or not of the same data type
>	Greater than	Returns TRUE if the left operand is greater than the right operand
<	Less than	Returns TRUE if the left operand is less than the right operand
>=	Greater than or equal to	Returns TRUE if the left operand is greater than or equal to the right operand
<=	Less than or equal to	Returns TRUE if the left operand is less than or equal to the right operand

Logical Operators

Symbol	Operation	Description
&& or AND	Logical And	Returns TRUE if both the left operand and right operand return a value of TRUE; otherwise, it returns a value of FALSE
]] or OR	Logical Or	Returns TRUE if both the left operand or right operand return a value of TRUE; otherwise, (neither operand returns a value of TRUE), it returns a value of FALSE
XOR	Logical Exclusive Or	Returns TRUE if only one of the left operand or right operand returns a value of TRUE; otherwise (neither operand returns a value of TRUE or both operands return a value of TRUE), it returns a value of FALSE
!	Logical Not	Returns TRUE if an expression is FALSE and returns FALSE if an expression is TRUE

- Example:

```
$Gender = “male”;
$Age = 17;
$RiskFactor = $Gender ==”male” && $Age <=21;   //returns TRUE
```

- **Bitwise Operator**
  - **“&”** - This is the “Bitwise AND” operator. It will place 1 in each position where both operands have 1.
  - **“|”** - This operator, known as Bitwise OR, places 1 in positions where at least one operand has 1.
  - **“^”** - Programmers refer to this as the “Bitwise XOR” operator. It places 1 in each position where only one of the operands has 1.

Unary Operator

- **Negation Operators**
- **“!”** - Programmers call this the “Logical Negation” operator. It will give you true if the operand’s value is false. If the value is true, on the other hand, this operator will give you false.
- **“~”** - This is the Bitwise Negation operator. It replaces 0 with 1, and vice versa.

- **Cast Operators**
  - (array) – This operator changes the data type of a value to “array.”
  - (int) or (integer) – Use this operator to convert values into integers.
  - (string) – This is the operator that you should use to create string values out of nonstring ones.

- (object) – With this operator, you can tag any value as an “object.”
- (real), (float) or (double) – This operator allows you to convert values from any data type into floating-point values.
- (bool) or (boolean) – Use this operator to convert any value into its Boolean form.

Example:

```
$sample_string “10”;
$sample_number = (int) $sample_string;
```

Ternary Operator

- **"?"** operator - evaluates the result of "conditional_statement." If the result is true, the operator will return the value of "first_expression." If the result is false, the operator will give you the value of "second_expression."
- Syntax:

conditional_statement ? first_expression : second_expression

Type Casting

- **Syntax:**

  $NewVariable = *(new_type)* $oldVariable;

- *(new_type)* – is the type-casting operator representing the type to which you want to cast the variable.
- Example:

```
$SpeedLimitMiles = "55 mph";
$speeLimitKilometers = (int) $SpeedLimitMiles *
1.6;
echo "$SpeedLimitMiles is equal to
$speeLimitKilometers kph";
```

- Using *gettype()*
- Boolean
- Integer
- Double
- String
- Array
- Object
- Resource
- NULL
- Unknown type

- Syntax:

  gettype*($variable_name)*;

- Example:

  $MortgageRate = .0575;

  echo gettype($MortgageRate);

Understanding Operator Precedence

Symbol	Operator	Associativity
new clone	New object-highest precedence	None
[ ]	Array elements	Right to left
++ –	Increment/Decrement	Right to left
(int) (double) (string) (array) (object)	Cast	Right to left
@	Suppress errors	Right to left
instance of	Types	None
!	Logical Not	Right to left
* / %	Multiplication/division/modulus	Left to right
+ - .	Addition/subtraction/string concatenation	Left to right

Symbol	Operator	Associativity
< <= > >= <>	Comparison	None
== != === !==	Equality	None
&&	Logical And	Left to right
||	Logical Or	Left to right
? :	Conditional	Left to right
= += -= *= /= %=	Assignment	Right to left
AND	Logical And	Left to right
XOR	Logical Exclusive Or	Left to right
OR	Logical Or	Left to right
,	List separator-lowest precedence	Left to right

- Acknowledgement:
- www.freepik.com
- www.codeauthority.com
- www.vectorsctock.com

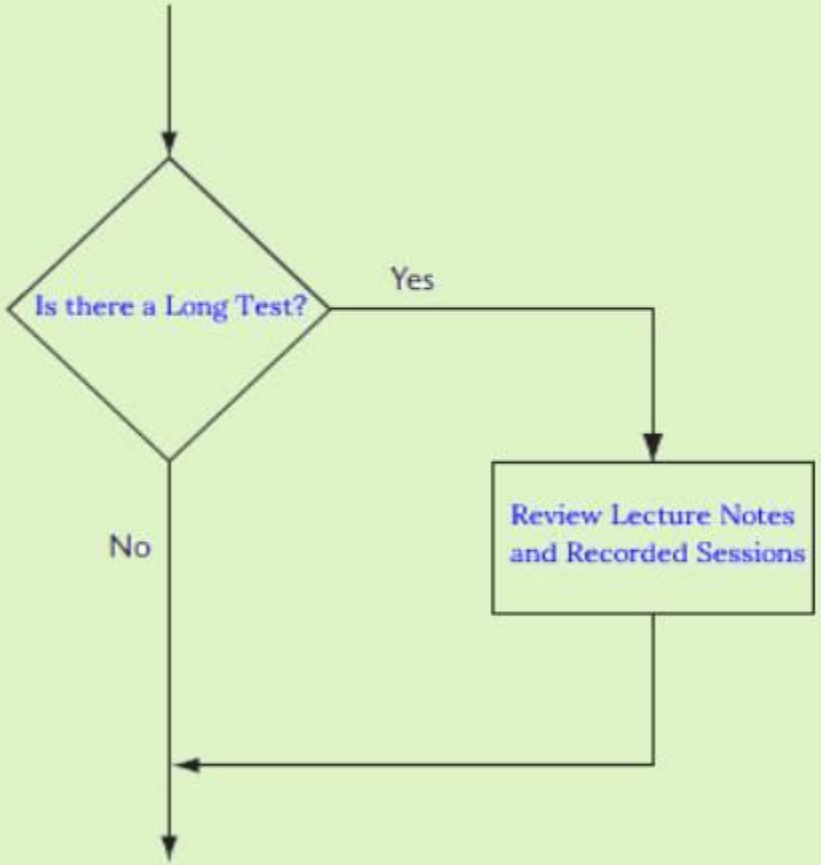
PHP:
Selection Constructs

Bernard G. Sanidad, DBA, MIT

Learning Objectives

- At the end of the discussion, Students would be able to:
  - Discuss the different selection controls in PHP;
  - Apply the correct syntax for each selection controls;
  - Analyze the application of selection control in providing solution to problems from various fields;
  - Create a script that reflect topics learned from the discussion;

The Decision Structure in Flowchart



- Decision making or flow control is the process of determining the order in which statements execute in a program
- The special types of PHP statements used for making decisions are called decision-making statements or decision-making structures

If statement

- Used to execute specific programming code if the evaluation of a conditional expression returns a value of TRUE
- Syntax:

if (conditional expression)

statement;

```php
<?php
    $ExampleVar = 18;

    if ($ExampleVar >= 18);
        echo "<p> The age is allowed to vote. </p>";
        echo "<p>The text generated after the 'if' statement </p>";
?>
```

```php
<?php
    $variable = 4;

    if ($variable == 5)
        echo "<p> This text will not appear. </p>";
    echo "<p> This is the only text that appears.</p>";
?>
```

```php
<?php

    $variable = 6;

    if ($variable == 5)
    {
        echo "<p> The condition evaluates to true. </p>";
        echo "<p> $variable is equal to " .$variable. ". </p>";
        echo "<p> Each of these lines will be displayed. </p>";
    }
    echo "<p> This statement always executes after the 'if' statement. </p>";

?>
```

if…else Statement

- An if statement that includes an else clause is called an **if...else statement**
- An else clause executes when the condition in an if...else statement evaluates to FALSE
- Syntax:

```
if(conditional expression)
        statement;
else
        statement;
```

```php
<?php
    $sex = "F";

    if ($sex =="F")
    {
        echo "<p> Female </p>";

    }
    else
    {
        echo "<p> Male </p>";
    }
?>
```

if…elseif…else statement

- The if…elseif…else statement executes different codes for more that two conditions.
- Syntax:

```
if(conditional expression1)
        statement;
elseif(conditional expression2)
        statement;
elseif(conditional expression3)
        statement;
…
else
        statement;
```

```php
<?php
    $age = 55;

    if ($age < 21)
    {
        echo "<p> Your age is less than 21. </p>";
    }
    elseif ($age > 21 && $age < 30)
    {
        echo "<p> Your age is between 22 and 29. </p>";
    }
    else
    {
        echo "<p> Undefined age. </p>";
    }

?>
```

Switch Statement

- The same variable is compared with many different values
- The default statement is used if none of the cases are true
- Statements are execute until it sees a break statement

- Syntax:

```
switch($variable_name)
        {
                case label:
                        statement;
                        break;
                case label:
                        statement;
                        break;
                default:
                        statement;
        }
```

Break

- Break statement ends execution of the current for, while, do-while or switch structure.
- Break accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of

```php
<?php

    $season = "winter";
    switch ($season)
    {
        case "spring":
            echo "<p> Flowers and trees beginning to bloom and weather beginning to warm up. </p>";
            break;
        case "summer":
            echo "<p> The warmest weather. The flower and trees are in full bloom. </p>";
            break;
        case "autumn":
            echo "<p> Weather beginning to cool and the leaves on the trees changing color. </p>";
            break;
        case "winter":
            echo "<p>Trees no longer have leaves . Most likely to snow in areas that get snow. </p>";
            break;
        default:
            echo "<p> Invalid input. </p>";
    }
?>
```

- Acknowledgement:
  - www.freepik.com
  - www.codeauthority.com
  - www.vectorsctock.com