PHP CONSTANTS AND OPERATORS

Presenter:
Agubata Odinaka

CONSTANTS

PHP CONSTANTS

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

PHP CONSTANTS

PHP Constants

There are two ways to define constants:

- By using the ‘const’ keyword:

Example:

```
const AMOUNT= “20”;
```

PHP CONSTANTS

- By using the define() function:

Syntax: define(“name” , “value”, “case-sensitive”);

Example:

define(“AMOUNT” , ”20”, “”);

PHP CONSTANTS

To define a constant you have to use define() function and to retrieve the value of a constant, you simply specify its name. You can also use the function constant() to read a constant's value if you wish to obtain the constant's name dynamically.

PHP CONSTANTS

Example:

```
<?php
define("MINSIZE", 50);
echo MINSIZE;
echo constant("MINSIZE"); // same thing as the previous line
?>
```

PHP CONSTANTS

Constants may be defined and accessed anywhere without regard to variable scoping rules. i.e. constants are automatically global across the entire script.

PHP CONSTANTS

Example:

```
<?php
define("GREETING", "Hello world");
function myTest() {
    echo GREETING;
}
myTest();
?>
```

OPERATORS

What is an Operator

In computer programming an operator is a character that represents an action, using expression *4 + 5 is equal to 9*.

Operators are used to perform operations on variables and values.

Here 4 and 5 are called operands and + is called operator or arithmetic operator that represents addition.

Operator Precedence

PHP language supports the following type of operators.

- Arithmetic Operators
- Increment/Decrement Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Operator Precedence

The precedence of an operator specifies how "tightly" it binds two expressions together.

For example, in the expression *1 + 5 * 3*, the answer is *16* and not *18* because the multiplication ("*") operator has a higher precedence than the addition ("+") operator.

Operator Precedence

Parentheses may be used to force precedence, if necessary. For instance: *(1 + 5) * 3* evaluates to *18*. If operator precedence is equal, left to right associativity is used.

Arithmetic Operators

-$a =>	Negation =>Opposite of $a.
$a + $b =>	Addition => Sum of $a and $b.
$a - $b =>	Subtraction => Difference of $a and $b.
$a * $b =>	Multiplication => Product of $a and $b.
$a / $b =>	Division => Quotient of $a and $b.
$a % $b =>	Modulus => Remainder of $a divided by $b.
$a**$b=>	Result of raising $a to the $b'th power.

Assignment Operators

The basic assignment operator is "=". Your first inclination might be to think of this as "equal to". Don't. It really means that the left operand gets set to the value of the expression on the rights (that is, "gets set to").

The value of an assignment expression is the value assigned. That is, the value of "$a = 3" is 3.

Assignment Operators

$a=$b => $a=$b	$a is equal to $b
$a + =$b =>$a=$a+$b	$a is Sum of $a and $b.
$a - =$b =>$a=$a-$b	$a is Difference of $a and $b.
$a *=$b =>$a=$a*$b	$a is Product of $a and $b.
$a /=$b =>$a=$a/$b	$a is quotient of $a and $b.
$a %=$b =>$a=$a%$b	$a is Modulus of $a and $b.

Increment/Decrement Operators

++$a => Pre-increment => Increments $a by one, then returns $a.

$a++ => Post-increment => returns $a, then increments $a by one.

--$a => Pre-decrement => decrements $a by one, then returns $a.

$a-- => Post-decrement => returns $a, then decrements $a by one.

Comparison Operators

$a == $b =>	Equal
$a === $b =>	Identical
$a != $b =>	Not equal
$a !== $b =>	Not identical
$a < $b =>	Less than
$a > $b =>	Greater than
$a <= $b =>	Less than or equal to
$a >= $b =>	Greater than or equal to

Incrementing/Decrementing Operators

```
<?php
$a = 5;
//because of $a before ++,it will return $a value before it returns
$a++ increment
echo $a++ ."<br>";
echo $a;
?>
```

Logical Operators

$a and $b => AND =>TRUE if both $a and $b are TRUE.

$a or $b => OR =>TRUE if either $a or $b is TRUE.

$a xor $b => XOR => TRUE if either $a or $b is TRUE, but not both.

! $a => NOT TRUE => TRUE if $a is not TRUE.

$a && $b => AND => And TRUE if both $a and $b are TRUE.

$a || $b => Or => TRUE if either $a or $b is TRUE.

If $username==‘john‘ && $password =='ben123' || $username'

=='chike'&& $password' =='chike123'

String Operators

There are two string operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments.

Example:

```
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!“
Echo $b;
Echo “<br>”;
```

String Operators

The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side.

```
$a = "Hello ";
$a .= "World!";     // now $a contains "Hello World!“
echo $a;
```

Questions?