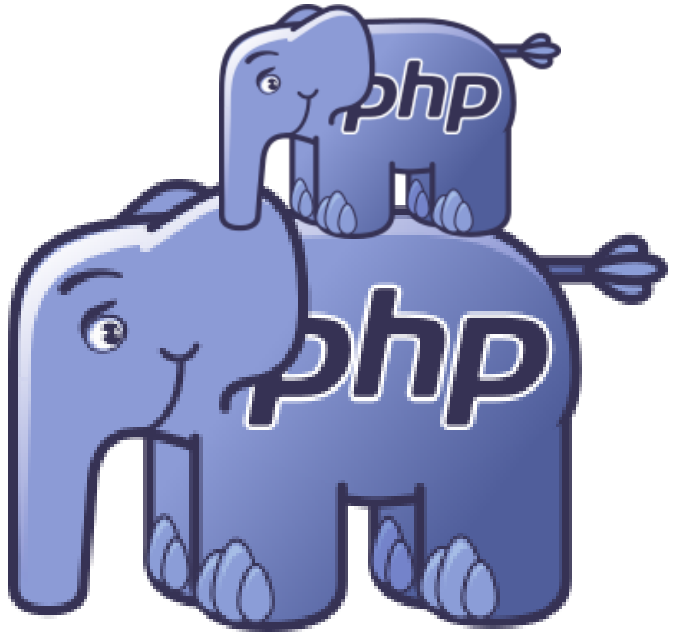
PHP Basics



OPERATORS

FLOW-CONTROL STATEMENTS

EASTERN WASHINGTON UNIVERSITY

Operators

Based on C and Perl

- Arithmetic
- Logical, bit-wise, shift
- Subscript
- Assignment with operation
- Precedence as expected
  - Can use ( ) to associate

Arithmetic Operators

Standard: `+ - * / % **`

Unary: `- +`

Autoincrement, autodecrement: `++ --`

- pre-increment: `++$n`
- post-increment: `$n++`

Can be Applied to Strings

Turns alphabetic strings to next letter in alphabet

- “a” -> “b”
- “z” -> “aa”
- “spaz” -> “spba”

Treats alphabetics as 26-value number system

String Concatenation Operator

(period) .

```
$n  = 5;
$s = ‘There are ’ . $n . ‘ ducks’;
echo $s;
```

Comparison Operators

Standard: `==` `!=` `<>` `>` `<` `>=` `<=`

Can compare:

- Number to Number
- String to Number
- String to String

Two numeric strings are compared as if numbers

- Use `strcmp()` to compare lexicographically

Comparison Operators

Identity: `===` (true if same type and same value)

Not Identical: `!==`

Spaceship: `<=>`

- Darth Vader's TIE Fighter
  - `0` if equal
  - `-1` if left < right
  - `+1` if left > right

Null Coalescing Operator

(double question mark) ??

- If left is NULL, evaluate to right

```
$var1 = NULL;
$var2 = 31;
echo $var1 ?? $var2; // outputs 31
```

Logical Operators

AND: `&&` `and`

OR: `||` `or`

XOR: `xor`

Negation: `!`

Bitwise Operators

AND: &

OR: |

XOR: ^

Negation: ~

Shifts: >> <<

- Right value is number of bits to shift left value
- Convert to integer; right shift: add '0' if positive, '1' if negative

Casting Operators

`(int)`	to integer
`(bool)`	to Boolean
`(float)`	to floating-point
`(string)`	to string
`(array)`	to array
`(object)`	to object

```
$a = “5”;
$b = (int) $a;
echo $b, “\n”;
echo $a, “\n”;
```

Casting object into array

Builds an array of properties

```
$ed = new Person;
$ed->set_name('Edison');
echo "Hello, $ed->name \n";

$name_arr = (array) $ed;
```

Casting array into object

Builds an object whose properties correspond to the array keys, values

```
$spies = array('Tinker' => "Percy",
               'Baker' => "Tom");

$traitors = (object) $spies;
echo $traitors->Tinker;
```

Assignment Operator

Standard: `=` (duh)

With operation:

- Arithmetic: `+=` `...`
- Bit-wise: `&=` `...`
- Concatenate: `.=`

Conditional Operator (ternary)

Test Boolean condition and return value: `?:`

```
$B ? $X : $Y
```

- If `$B` evaluates to `true`, return `$X`
- If `$B` evaluates to `false`, return `$Y`

Type Test Operator

Keyword: `instanceof`

```
$ed = new Person;
. . .
$isaPerson = $ed instanceof Person;  //true
```

Flow-Control Statements

Conditional Statements

`if` -- checks truthfulness of expression, evaluates statements if `true`

```
if (expression) statement
```

`if-else` -- adds statement for execution if false

```
if (expression)
  statement
else statement
```

Conditional Examples

```
$a = “5”;
$b = 5;
if ($a == $b) {
    $c = 10;
    echo “Same\n”;
}
else {
    echo “Not Same\n”;
    $c = 12;
}
```

Alternate Syntax

```
$a = “5”;
$b = 5;
if ($a == $b) :
    $c = 10;
    echo “Same\n”;
else:
    echo “Not Same\n”;
    $c = 12;
endif
```

Why Alternate Syntax?

```
<?php if ($user_validated) : ?>
 <table>
   <tr>
     <td>First Name:</td> <td>Sophia</td>
   </tr>
   <tr>
     <td>Last Name:</td> <td>Lee</td>
   </tr>
 </table>
<?php else: ?>
 Please log in.
<?php endif ?>
```

Nested Conditional Statements

`if-elseif-else` -- nested conditionals

```
if (expression)
  statement
elseif (expression)
  statement
. . .
else
  statement
```

Switch Statement

Like nested conditionals

```
switch (variable) {
  case value :
    statement
  break;
  case value :
    statement
  break;
   . . .
  default :
    statement
  break;
}
```

Alternate Syntax

```
switch (variable) :
  case value :
    statement
  break;
  case value :
    statement
  break;
   . . .
  default :
    statement
  break;
endswitch;
```

While Loop

Loop while expression is `true`

```
while (expression)
  statement
```

While Loop Examples

```php
$total = 0;
$i = 1;
while ($i <= 10) {
    $total += $i;
    ++$i;
}
echo $total;
```

Alternate Syntax

```php
$total = 0;
$i = 1;
while ($i <= 10) :
    $total += $i;
    ++$i;
endwhile;
echo $total;
```

Looping Controls

`continue` – jump to next test in loop

`break` – breaks out of loop

- Can take an integer value to jump out of nested loops

Do-While Loop

Loop while expression is `true`

- Executes loop body at least once

```
do
  statement
while (expression);
```

```
$total = 0;
$i = 1;
do
  $total += $i++;
while ($i <= 10);

echo $total;
```

For Loop

Loop over counter values

```
for (start; condition; increment)
  statement
```

For Loop Example

```php
$total = 0;
for ($i = 1; $i <= 10; ++$i)
  $total += $i;

echo $total;
```

Alternate Syntax

```php
$total = 0;
for ($i = 1; $i <= 10; ++$i) :
  $total += $i;
endfor;

echo $total;
```

EASTERN WASHINGTON UNIVERSITY

Foreach Loop

Iterate over elements of an array

```
foreach (array as name)
  statement
```

```
foreach (array as key => value)
  statement
```

Foreach Examples

```
foreach ($spy as $code_name) {
    echo “Who is {$code_name}?\n”;
}
```

```
foreach ($name as $code => $person) {
    echo “$code is designated to {$person}\n”;
}
```

Foreach Examples – Alternate Syntax

```
foreach ($spy as $code_name) :
    echo “Who is {$code_name}?\n”;
endforeach;
```

```
foreach ($name as $code => $person) :
    echo “$code is designated to {$person}\n”;
endforeach;
```

Miscellaneous

`try..catch` – graceful way to handle errors

`exit` – ends script's execution

`return` – return from function or script