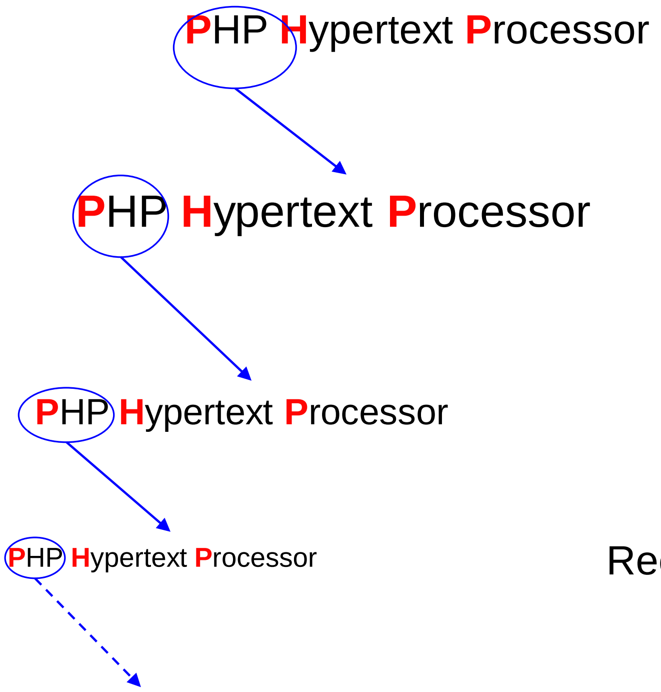
What does **PHP** stand for?



Recursive acronym!

Not Personal Home Page

What is **PHP**?

- Server-side programming language

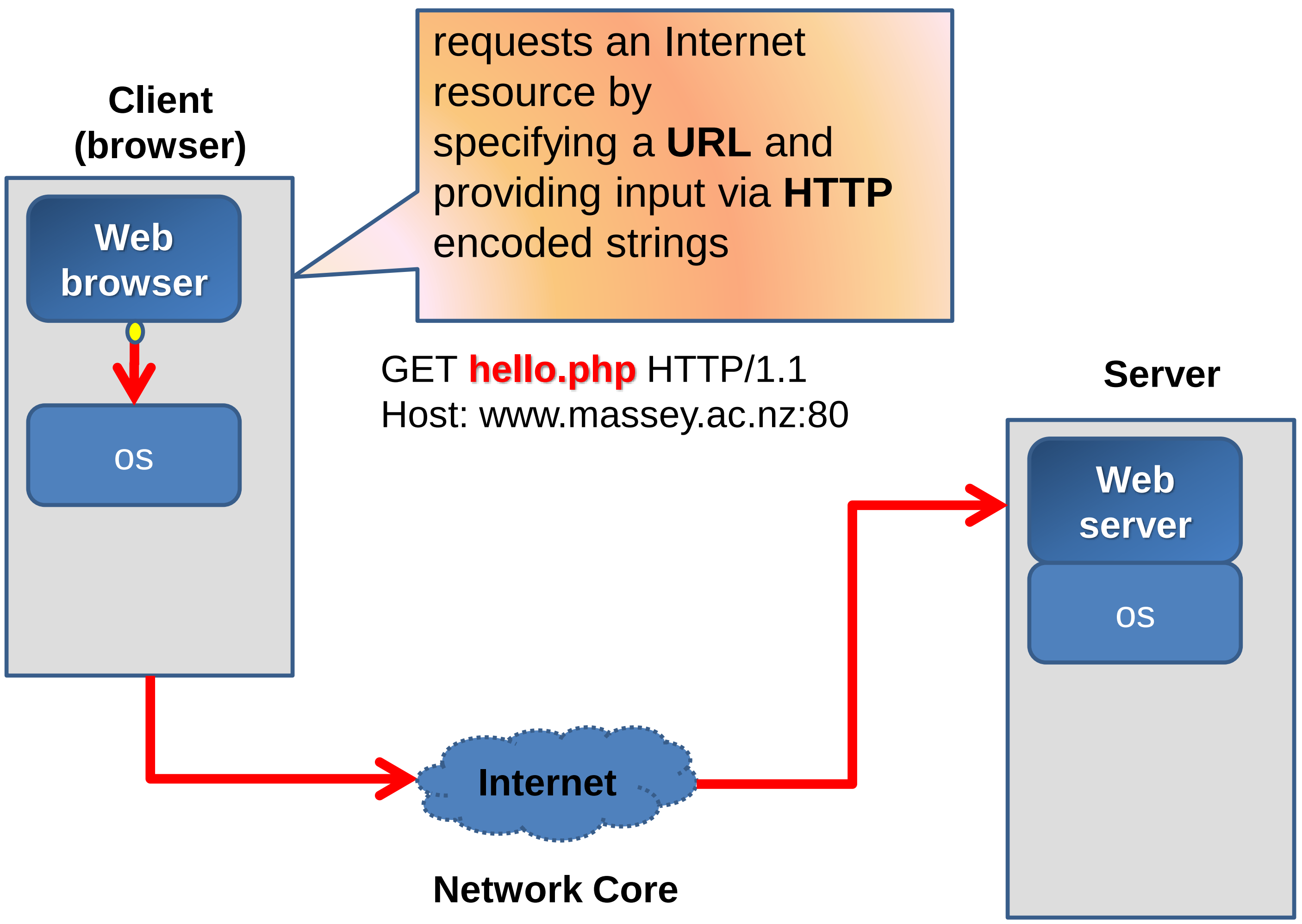
- Designed for quick development of HTML based dynamic web pages
  - Server side scripts embedded in HTML pages
  - **<?php** …… **?>**

- Elements of C, Java, and Perl

Evolution of PHP

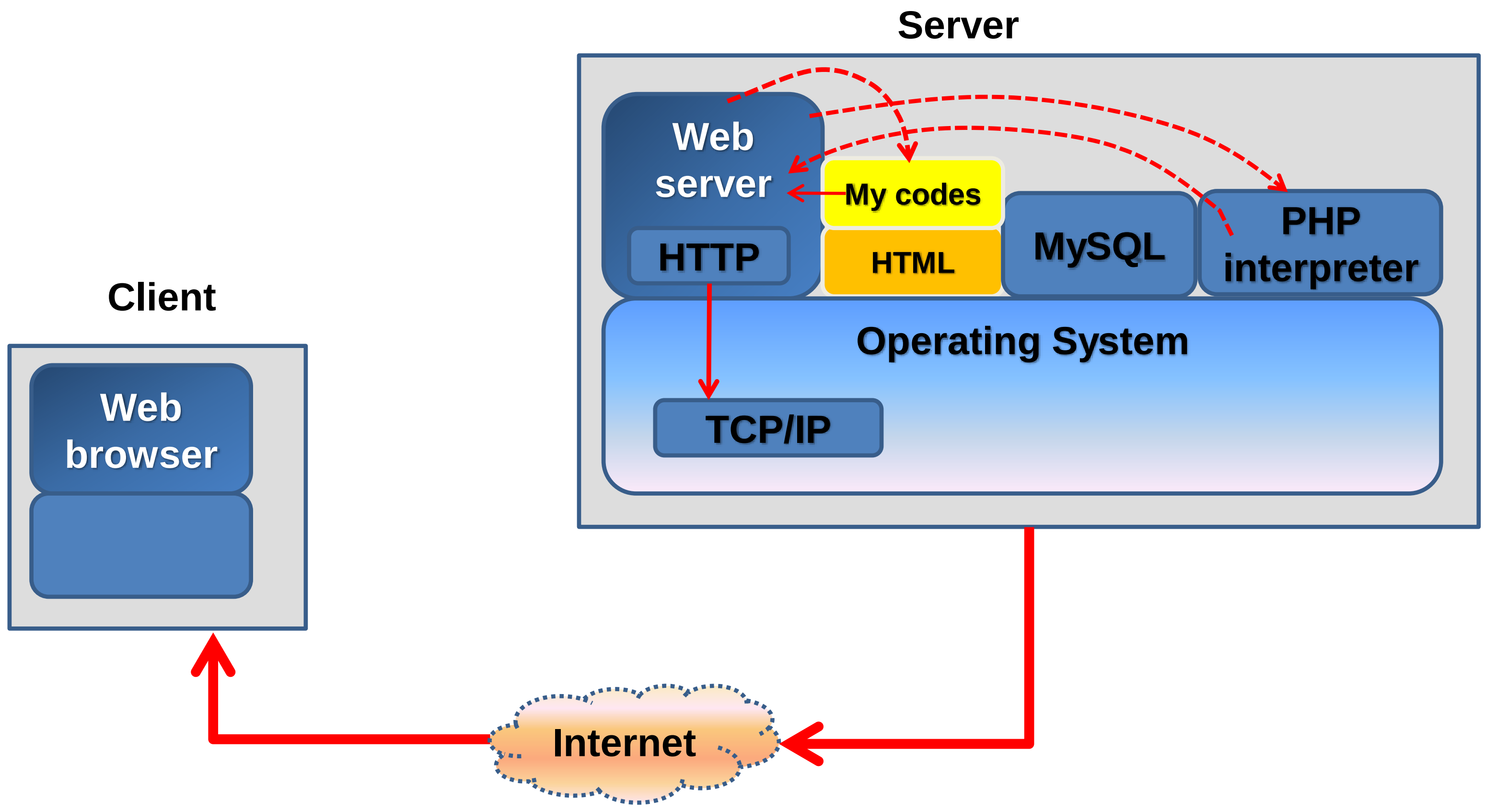
- 1994, *Rasmus Lerdorf*: started off as a series of Perl scripts
- June 8, 1995: Personal Home Page/Forms Interpreter, released to the public
- June, 1998, PHP3
- PHP4 introduced features of object-oriented programming
  - Objects/classes
  - Pass/return by reference
- PHP5 added more OO features
  - Exception handling
  - Abstract classes and methods

Client: makes a request

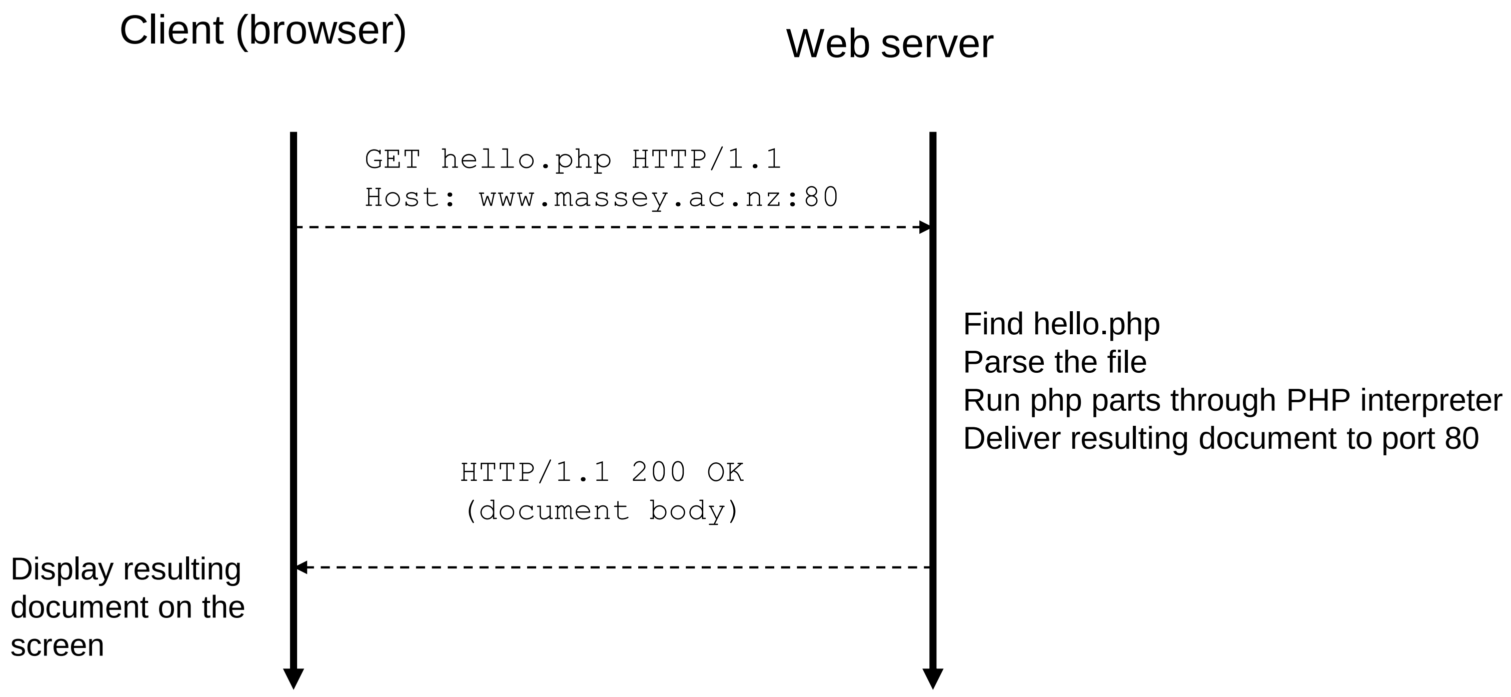

Client
(browser)
Web browser
OS
requests an Internet resource by specifying a **URL** and providing input via **HTTP** encoded strings
GET **hello.php** HTTP/1.1
Host: www.massey.ac.nz:80
Server
Web server
OS
Internet
Network Core

Server: responds

- Webserver supports HTTP.



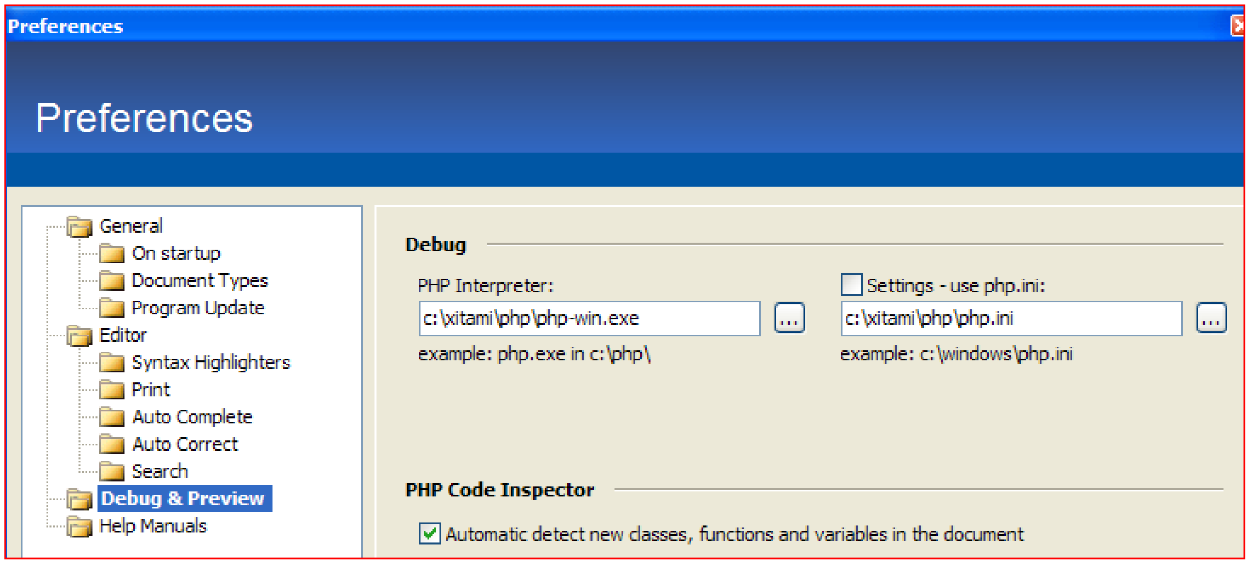
What's happening?


Client (browser)
Web server
GET hello.php HTTP/1.1
Host: www.massey.ac.nz:80
Find hello.php
Parse the file
Run php parts through PHP interpreter
Deliver resulting document to port 80
HTTP/1.1 200 OK
(document body)
Display resulting document on the screen

PHP Designer

Configure your PHP editor to access the PHP interpreter

From the menu bar, select **Tools/Preferences**



Debug

PHP Interpreter: c:\xitami\php\php-win.exe

example: php.exe in c:\php\

☐ Settings - use php.ini: c:\xitami\php\php.ini

example: c:\windows\php.ini

PHP Code Inspector

☑ Automatic detect new classes, functions and variables in the document

1st PHP program

Enter this text in a file called “hello.php”

```
<html>
<head><title>Hello</title></head>
<body>

<?php
    print "<h1>Hi there!</h1>";
?>

</body>
</html>
```

Load the URL in a browser and see what happens

Comparison with Javascript

Javascript is sent from server and runs on the client side.

PHP runs on the server, just like **CGI** scripts.
The client does not see the PHP code - **only the results**.

But PHP has some advantages.

Interpreted language

PHP is an interpreted language, i.e. an interpreter runs the code directly without compiling

Interpreted languages include: **PHP, Perl, Python**

Compiled languages include: **C, C++, Java**

PHP code is **executed** on the **server**, and the plain **HTML result** is sent to the **browser.**

Basic Syntax

• a PHP scripting block always starts with **<?php** and ends with **?>**.
• a PHP scripting block can be placed anywhere in the document.

• contains **HTML tags**, just like an HTML file, and some PHP scripting code.

• each code line in PHP must end with a **semicolon**.

• The file must have a **.php extension**. If the file has a .**html** extension, the PHP code will not be executed.

- possible extensions: "**.php**", "**.php3**", "**.php4**", "**.php5**“, "**.phtml**”,
- "**.inc**", "**.tpl**",

PHP Variables

Variables in PHP are **dynamically typed** - no need to declare them
 - PHP is a weakly-typed programming language

Variable names begin with **$** then a character (**not number**)

✓ **$value** = 1;

✓ **$x** = 1.432344;

✓ **$myName** = “Rasmus Lerdorf”;

✗ **yourName** = “Zeev Suraski”; **//invalid, missing $**

✗ **$7eleven** = “McGyver”; **//invalid, starts with a number**

PHP supports **references** (BE CAREFUL syntax is slightly different from C++!)

```
$a = &$x;
```

Assign by reference (an alias)

To assign by reference, simply place an ampersand (**&**) at the beginning of the variable which is being assigned (the source variable). For instance, the following code snippet outputs '*My name is Angus*' twice:

```
<?php

$foo = ‘Angus';                  // Assign the value ‘Angus' to $foo
$bar = &$foo;                    // Reference $foo via $bar.
$bar = "My name is $bar";  // Alter $bar...
echo $bar;
echo $foo;                  // $foo is altered too.

?>
```

Try the following

Modify “hello.php” to print your name after assigning it to a variable.

What happens if this line is inserted into the php script?

```
print "<small>" . date("Y M d", getlastmod()) . "</small>";
```

What happens if you call **phpinfo()**?

phpinfo() is a valuable debugging tool as it contains all EGPCS (Environment, GET, POST, Cookie, Server) data.

Jumping in and out of PHP

```
<html><head><title>1st PHP</title></head><body>

    <?php
    print "This line was PHP generated";
    ?>
    <p> This line was not... </p>
    <?php
    print "This line was also PHP generated";
    ?>

</body></html>
```

Comments

C-style:

```
// this is a comment
```

Shell script style:

```
# this is a comment
```

Multi-line comments:

```
/* this is a comment
this is the continuation of the comment */
```

Note: Nested multi-line comments are not allowed.

PHP Variables

Boolean	**boo**
Integer	**int**
Float	**flo**
String	**str**
Array	**arr**
Object	**obj**

PHP Variables

Convention:

```
$strName1='some name';
$intNumber1=2000000000;
$floNumber2=2.456;
$floNumber3=1.8e308; //notice scientific notation
```

PHP Constants

By default, case-sensitive as are variables

```
define(“DEFAULT_SCRIPT”, “php”);
define(“MIN_SPEED”, 2);
```

```
define(“DEFAULT_SCRIPT”, “php”,TRUE);
```

You can turn a constant variable case-insensitve
using a third argument set to TRUE.

PHP Operators

=, ==, +, -, /, *, &&, ||, !, ++, --, %, /=, *=, >, <, >=, <=, &, |, ^, ~

– **All similar to C**

Additionally,

.	**String concatenation** $fullName = $firstName . " " . $lastName;
===, !===	**Identical test operator: same type** as well **as same value**
@	**Error suppression command** When placed in front of an expression in PHP, any error messages that might be generated by that expression will be ignored.
``	**Back tick** to execute **shell commands** (be careful for variations on different platforms)

Execution Operator (` `)

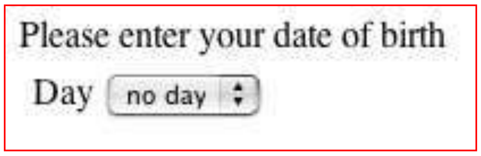
Note that these are not single-quotes!

PHP will attempt to execute the contents of the **backticks** as a shell command;

```
<?php
  $output = `dir *.php`;
  echo "<pre>$output</pre>";
?>
```

HTML with Form

Consider this html source to produce a simple form.


Please enter your date of birth

Day no day

Very cumbersome and error prone!

```
<html>
<head><title>Date of Birth</title></head>
<body>
<p>Please enter your date of birth </p>
<form action="dataEntry.php" method="post">
  <table cellspacing="5">
    <tr><td valign="middle" colspan=2> Day
    <!-- drop-down list for days -->
    <select name=day>
                        <option selected="selected"  value=""> no day </option>
                         <option value="1">1</option>
                        <option value="2">2</option>
                        <option value="3">3</option>
                        <option value="4">4</option>
                        <option value="5">5</option>
                        <option value="6">6</option>
                        <option value="7">7</option>
                        <option value="8">8</option>
                        <option value="9">9</option>
                        <option value="10">10</option>
                        <option value="11">11</option>
                        <option value="12">12</option>
                        <option value="13">13</option>
                        <option value="14">14</option>
                        <option value="15">15</option>
                        <option value="16">16</option>
                        <option value="17">17</option>
                        <option value="18">18</option>
                        <option value="19">19</option>
                        <option value="20">20</option>
                        <option value="21">21</option>
                        <option value="22">22</option>
                        <option value="23">23</option>
                        <option value="24">24</option>
                        <option value="25">25</option>
                        <option value="26">26</option>
                        <option value="27">27</option>
                        <option value="28">28</option>
                        <option value="29">29</option>
                        <option value="30">30</option>
                        <option value="31">31</option>
         </select>
     </td></tr>
   </table>
</form> </body> </html>
```

PHP makes it easier

```
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
<option value="13">13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20">20</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="23">23</option>
<option value="24">24</option>
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>
<option value="31">31</option>
```

```
<?php
  for ($day = 1; $day<= 31; ++$day) {
       print "\t<option value=\"$day\">$day</option>\n";
  }
?>
```

Note: use the escape sequence **\”** to produce double quotes

Escape Characters

\n	line feed character
\r	carriage return character
\t	tab
\	backslash character
\$	dollar sign
\”	double quote character

Arrays

An array in PHP is actually an ordered map.
A map is a type that associates ***values*** to ***keys***.

```
array( key => value , ... )

// key may only be an integer or string
// value may be any value of any type
```

Creating & printing Arrays

```php
// Create a simple array.
$array = array(1, 2, 3, 4, 5);
print_r($array);
```

Output:

```
Array (
[0] => 1
[1] => 2
[2] => 3
[3] => 4
[4] => 5
)
```

Creating & printing Arrays

```
// Create an array with forced indices.
$array = array(1, 1, 1, 1,  1, 8 => 1,  4 => 1, 19, 3 => 13);
print_r($array);
```

Output:

```
Array (
[0] => 1
[1] => 1
[2] => 1
[3] => 13
[4] => 1
[8] => 1
[9] => 19
)
```

Type of indices may vary within the same array

```
<?php
$arr = array(“candy" => "bar", 12 => true);

echo $arr[“candy"];  // bar
echo $arr[12];       // 1
?>
```

Note: Attempting to access an array key which has not been defined is the same as accessing any other undefined variable: an **E_NOTICE**-level error message will be issued, and the result will be **NULL**.

Auto-indexing

```php
<?php
// This array is the same as ...
array(5 => 43, 32, 56, "b" => 12);

// ...this array
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Printing Array Elements

```
<?php
  $array = array(5=>43, 32, 56, "b"=> 12);
  print("\$array[\"b\"]="); ✓
  echo $array["b"]; ✓
  print "<br>"; ✓
  print("\$array[\"b\"]= $array['b']");   ✗
  print("\$array[\"b\"]= {$array['b']}"); ✓
  print("\$array[\"b\"]= {$array[“b”]}"); ✓
  print("\$array[\"b\"]= $array["b"]"); ✗
?>
```

Output of the correct statements:

```
$array["b"]=12
```

More Print Examples

```php
<?php
	print "<br>";
	print("Hello World<br>");
	print "Hello World<br>";  //print() also works without parentheses
	$number=7;
	print $number;  //you can just print variables without double quotes
	print '<br>$number';  //this will print the variable name.
?>
```

Output:

```
Hello World
Hello World
7
$number
```

Extending & Modifying Arrays

```
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56;    // This is the same as $arr[13] = 56;
               // at this point of the script

$arr["x"] = 42; // This adds a new element to
                // the array with key "x"

unset($arr[5]); // This removes the element from the array

unset($arr);    // This deletes the whole array
?>
```

int array_unshift()

Prepend one or more elements to the beginning of an array

Note that the list of elements is prepended as a whole, so that the prepended elements stay in the same order. All numerical array keys will be modified to start counting from zero while literal keys won't be touched.
-returns the **new** number of elements.

```
<?php
$queue = array("orange", "banana");
array_unshift($queue, "apple", "raspberry"
);
print_r($queue);
?>
```

```
Array (
 [0] => apple
 [1] => raspberry
 [2] => orange
 [3] => banana
)
```

int **array_push()**

Push one or more elements onto the end of array.
Returns the new number of elements

```
<?php
$stack = array("orange", "banana");
array_push($stack, "apple", "raspberry");
print_r($stack);
?>
```

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
    [3] => raspberry
)
```

mixed **array_pop()**

pops and returns the last value of the array, shortening the array by one element. If array is empty (or is not an array), NULL will be returned. Will additionally produce a Warning when called on a non-array.

```
<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_pop($stack);
print_r($stack);
?>
```

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
)
```

Iterating through Arrays

```
foreach (array_expression as $value) {
    statement
}
```

Example:

```
<html>
<body>
<?php

$x=array(“red",“green",“blue");
foreach ($x as $value) {

    echo $value . "<br />";
 }

?>

</body>
</html>
```

- Loops over the array given by *array_expression*.
- On each loop, the value of the current element is assigned to *$value* and the internal array pointer is advanced by one (so on the next loop, you'll be looking at the next element).
- Note: acts on a copy of the array

Output:

```
red
green
blue
```

Iterating through Arrays

foreach (array_expression as **$key** => **$value**)
 statement

Example:

```
<html>
<body>
<?php

$numbers = array("one"=>"une", "two"=>"deux",
"three"=>"trois", "four"=>"quatre", "five"=>"cinq");

foreach($numbers as $key=>$val) {
  print "English: " . $key . ", French " . $val . "<br/>";
}

?>

</body>
</html>
```

- Loops over the array given by *array_expression*.
- On each loop, the current element’s key is assigned to the variable *$key*.
- Note: acts on a copy of the array

Output:

```
English: one, French une
English: two, French deux
English: three, French trois
English: four, French quatre
English: five, French cinq
```

PHP Control Structures

- if, else, switch … case …, break, continue
  - Similar to C, but must use brackets {} always
- elseif
  - Different syntax to C, same semantics
- While loop …, do … while loop, for loop…
  - All similar to C

PHP Control Structures

```
switch(expression){
   case $Example:  //variable name
                  statements;
                  break;
   case "text":
                      //…
   case 75:
                     //…
   case -123.45:
                    //…
    default:
                   //…
}
```

Summary

- PHP runs in the web server
- The server creates HTML dynamically
- It is more efficient than using CGIs
- Variables
- Arrays