PHP MATRIAL BY

KARTIK JOSHI

Preview from Notesale.co.uk
Page 1 of 261

- An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly, such as when a style is to be applied to a single occurrence of an element.
- To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property.
- The example shows how to change the color and the left margin of a paragraph:

```
<p style="color: red; margin-left: 20px">
        This is a paragraph
</p>
```

⌘ Multiple Style Sheets:

- If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.
- For example, an external style sheet has these properties for the **h3** selector:

```
h3
{ color: red; text-align: left; font-size: 8pt }
```

And an internal style sheet has these properties for the **h3** selector:

```
h3
{ text-align: right; font-size: 20pt}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

color: red; text-align: right; font-size: 20pt

The color is inherited from the external style sheet and the text-alignment and the internal style sheet replaces the font-size.

Preview from Notesale.co.uk
Page 6 of 261

3) Class & ID:-

⌘ The Class Selector:

With the class selector you can define different styles for the same type of HTML element. Say that you would like to have two types of paragraphs in your document: one right-aligned paragraph, and one center-aligned paragraph. Here is how you can do it with styles:

```
p.right {text-align: right}
p.center {text-align: center}
```

color: red
}

- The style rule below will match any **p** element that has an **id** attribute with a value of "green":

p#green {color: green}

- The rule above will not match an **hl** element:

<h1 id="green">Some text data</h1>

⌘ Comments in CSS:

You can insert comments into CSS to explain your code, which can help you when you edit the source code at a later date. The browser will ignore a comment. A CSS comment begins with **"/*"**, and end with **"*/"**, like this:

```
/* This is a comment */
p
{
    text-align: right;
    /* This is comment. */
    color: red;
}
```

Preview from Notesale.co.uk
Page 8 of 261

4) CSS Font Property:-

⌘ CSS Font Properties:

- The CSS font properties allow you to change the font family, boldness, size, and the style of a text.

  - ✓ Font-family property
  - ✓ Font-style property
  - ✓ Font-weight property
  - ✓ Font-size property

  - ✓ **Font-family property:**

- The font-property is a prioritized list of font family names and/or generic family names for an element.
- The browser will use the first value it recognizes.
- Separate each value with a comma.

Value	Description

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>
```

Using an External JavaScript

Sometimes you might want to run the same JavaScript on several pages, without having to write the same script on every page.

To simplify this, you can write a JavaScript in an external file. Save the external JavaScript file with a .js file extension.

Note: The external script cannot contain the <script> tag!

To use the external script, point to the .js file in the "src" attribute of the <script> tag:

Preview from Notesale.co.uk
Page 28 of 261

```
<html>
<head>
<script src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

Note: Remember to place the script exactly where you normally would write the script!

JavaScript Comments

Comments can be added to explain the JavaScript, or to make it more readable.

Single line comments start with //.

JavaScript Multi-Line Comments

Multi line comments start with /* and end with */.

JavaScript Variables

As with algebra, JavaScript variables are used to hold values or expressions.

A variable can have a short name, like x, or a more descriptive name, like carname.

Rules for JavaScript variable names:

Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings:

The + operator can also be used to add string variables or text values together.
To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a very nice day".
To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:
"What a very nice day"

Adding Strings and Numbers:
Look at these examples:

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);
```

Preview from Notesale.co.uk
Page 30 of 261

```
    code to be executed if n is
    different from case 1 and 2
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

Looping Statements

Looping refers to the ability of a block of code to repeat itself. This repetition can be for a predefined number of times or it can go until certain conditions are met. For instance, a block of code needs to be executed till the value of a variable becomes 20 (Conditional Looping), or a block of code needs to be repeated 7 times.

For this purpose, Javascript offers 2 types of loop structures:

1) for Loops - This loop iterate a specific number of times as specified.
2) While Loops – This is Conditional Loops, which continue until a condition is met.

Preview from Notesale.co.uk
Page 34 of 261

For Loop

The **for** loop is the most basic type of loop and resembles a for loop in most other programming languages, including ANSI 'C'.

Syntax:

```
for(expression1; condition; expression2)
{
    // Javascript commands…
}
```

Where, expression1 sets up a counter variable and assigns the initial value. The declaration of the counter variable can also be done here itself, condition specifies the final value for the loop to fire (i.e. the loop fires till condition evaluates to true), expression2 specifies how the initial value in expression1 is incremented.

Example:

```
myCars[0]="Opel";
```

Now, the following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Opel
```

The Array Object
The Array object is used to store multiple values in a single variable.
Syntax for creating an Array object:

```
var myCars=new Array("Saab","Volvo","BMW")
```

To access and to set values inside an array, you must use the index numbers as follows:

- myCars[0] is the first element
- myCars[1] is the second element
- myCars[2] is the third element

Array Object Properties

Property	Description
constructor	Returns a reference to the array function that created the object
Index	
Input	
Length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to the object

Preview from Notesale.co.uk
Page 38 of 261

6. User Defined Function

Functions offer the ability to group together JavaScipt program code that performs a specific task into a single unit that can be used repeatedly whenever required in a Javascript program.
A user defined function first needs to be declared and coded. Once this is done the function can be invoked by calling it using the name given to the function when it was declared.
Functions can accept information in the form of arguments and can return results.

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software (OSS)
- PHP is free to download and use

✓ **What is a PHP File?**
- PHP files may contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

✓ **What is MySQL?**
- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

⇨ **PHP + MySQL**
- PHP combined with MySQL are cross-platform (means that you can develop in Windows and serve on a Unix platform)

✓ **Why PHP?**
- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

✓ **Where to Start?**
- Install an Apache server on a Windows or Linux machine
- Install PHP on a Windows or Linux machine
- Install MySQL on a Windows or Linux machine

❖ Advantages of PHP

1) Cost
- PHP costs you nothing. It is an open source software and doesn't need to purchase it for development.

2) Ease of Use

Preview from Notesale.co.uk
Page 59 of 261

```
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

4. Optionally add index.php to the DirectoryIndex directive to open index.php files if no file specified.

   ```
   DirectoryIndex index.html index.php
   ```

   Restart the **Apache-2.2** service.

   To test whether the PHP engine is installed successfully, create in the Apache HTTP server document root the "phpinfo.php" file with the following content:

   ```
   <?php
   phpinfo();
   ```

   Opening the "http://localhost:8081/phpinfo.php" URL in the browser should display the information about PHP configuration. Here 8081 is the port on which Apache HTTP sever listens for incoming requests. Replace it in case when the particular Apache service listens to the requests on different port.

Preview from Notesale.co.uk
Page 66 of 261

Configure IIS FastCGI

Following this instruction requires Array on your computer.

Create folders:

- C:WITSuiteConfigurationIISPHP-5.2
- C:WITSuiteTemporaryFilesIISPHP-5.2
- C:WITSuiteTemporaryFilesIISPHP-5.2wsdl
- C:WITSuiteTemporaryFilesIISPHP-5.2sessions
- C:WITSuiteTemporaryFilesIISPHP-5.2upload

Copy the C:WITSuiteLanguagesPHP-5.2php.ini-recommended to C:WITSuiteConfigurationIISPHP-5.2php.ini.

Change the following settings in the C:WITSuiteConfigurationIISPHP-5.2php.ini file:

PHP follows Perl's convention when dealing with arithmetic operations on character variables and not C's. For example, in Perl 'Z'+1 turns into 'AA', while in C 'Z'+1 turns into '[' (ord('Z') == 90, ord('[') == 91). Note that character variables can be incremented but not decremented and even so only plain ASCII characters (a-z and A-Z) are supported.

Example 1 Arithmetic Operations on Character Variables

```
<?php
$i = 'W';
for ($n=0; $n<6; $n++) {
   echo ++$i . "\n";
}
?>
```

The above example will output:

```
X
Y
Z
AA
AB
AC
```

Preview from Notesale.co.uk
Page 81 of 261

Incrementing or decrementing Booleans has no effect.

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Comparison Operators

Comparison operators, as their name implies, allow you to compare two values. You may also be interested in viewing the type comparison tables, as they show examples of various type related comparisons.

Operator Precedence		
Associativity	**Operators**	**Additional Information**
right	= += -= *= /= .= %= &= |= ^= <<= >>=	assignment
Left	and	logical
Left	xor	logical
Left	or	logical
Left	,	many uses

Left Associativity means that the expression is evaluated from left to right, right associativity means the opposite.

Example 1 Associativity

```
<?php
$a = 3 * 3 % 5; // (3 * 3) % 5 = 4
$a = true ? 0 : true ? 1 : 2; // (true ? 0 : true) ? 1 : 2 = 2

$a = 1;
$b = 2;
$a = $b += 3; // $a = ($b += 3) -> $a = 5, $b = 5
?>
```

Use parentheses to increase readability of the code.

Preview from Notesale.co.uk
Page 85 of 261

String Operators

There are two string operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side. Please read Assignment Operators for more information.

```
<?php
$a = "Hello

$b = $a . "World!"; // now $b contains "Hello World!"
$a = "Hello ";
$a .= "World!";     // now $a contains "Hello World!"
?>
```

```
/* example 2 */

$i = 1;
while ($i <= 10):
   echo $i;
   $i++;
endwhile;
?>
```

do-while

- *do-while* loops are very similar to *while* loops, except the truth expression is checked at the end of each iteration instead of in the beginning.
- The main difference from regular *while* loops is that the first iteration of a *do-while* loop is guaranteed to run (the truth expression is only checked at the end of the iteration), whereas it may not necessarily run with a regular *while* loop (the truth expression is checked at the beginning of each iteration, if it evaluates to **FALSE** right from the beginning, the loop execution would end immediately).

There is just one syntax for *do-while* loops:

```
<?php
$i = 0;
do{
   echo $i;
} while ($i > 0);
?>
```

Preview from Notesale.co.uk
Page 91 of 261

The above loop would run one time exactly, since after the first iteration, when truth expression is checked, it evaluates to **FALSE** ($i is not bigger than 0) and the loop execution ends.

Advanced C users may be familiar with a different usage of the *do-while* loop, to allow stopping execution in the middle of code blocks, by encapsulating them with *do-while* (0), and using the *break* statement. The following code fragment demonstrates this:

```
<?php
do{
 if ($i < 5) {
        echo "i is not big enough";
        break;
   }
```

```
/* example 3 */
$i = 1;
for (; ; ) {
    if ($i > 10) {
        break;
    }
    echo $i;
    $i++;
}
/* example 4  */
for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
?>
```

Preview from Notesale.co.uk
Page 93 of 261

foreach

- This simply gives an easy way to iterate over arrays. *foreach* works only on arrays, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable. There are two syntaxes; the second is a minor but useful extension of the first:

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

- The first form loops over the array given by *array_expression*. On each loop, the value of the current element is assigned to *$value* and the internal array pointer is advanced by one (so on the next loop, you'll be looking at the next element).
- The second form does the same thing, except that the current element's key will be assigned to the variable *$key* on each loop.

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 4
    [4] => 5
)
Array
(
)
Array
(
    [5] => 6
)
Array
(
    [0] => 6
    [1] => 7
)
```

if_else Example

Preview from Notesale.co.uk
Page 98 of 261

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
```

```
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>
</body>
</html>
```

The ElseIf Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

Preview from Notesale.co.uk
Page 99 of 261

The Switch Example

Example

```
<html>
<body>
<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
```

Function definition syntax

Function definitions have the following form:
function function-name ($argument-1, $argument-2, ..)
{
statement-1;
statement-2;
...
}

function definitions have four parts:

- The special word function
- The name that you want to give your function
- The function's parameter list—dollar-sign variables separated by commas
- The function body—a brace-enclosed set of statements

Function naming rule

- Just as with variable names, the name of the function must be made up of letters, numbers and underscores, and it must not start with a number.
- Unlike variable names, function names are converted to lowercase before they are stored internally by PHP, so a function is the same regardless of capitalization.

Preview from Notesale.co.uk
Page 101 of 261

Create a PHP Function

A function is a block of code that can be executed whenever we need it.

Creating PHP functions:

- All functions start with the word "function()"
- Name the function - It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
- Add a "{" - The function code starts after the opening curly brace
- Insert the function code
- Add a "}" - The function is finished by a closing curly brace

```
echo $sing;
echo $str;
?>
```

❖ chr

- Returns a one-character string containing the character specified by *ascii* .

Syntax:

string **chr** (int $ascii)

Example

```
echo chr(65);     //A
echo chr(97); //a
```

❖ ord

- Returns the ASCII value of the first character of *string* .

Syntax:

int **ord** (string $string)

Example:

```
Echo ord("A");    //65
Echo ord("abc"); //97
```

Preview from Notesale.co.uk
Page 118 of 261

❖ strtolower

- Returns *string* with all alphabetic characters converted to lowercase.

Syntax:

string **strtolower** (string $str)

Example :

```
echo strtolower("WelCome To HNS");     //  welcome  to hns
```

❖ strtoupper

- Find position of first occurrence of a string

Syntax:

int **strpos** (string $haystack , mixed $needle [, int $offset])

- Returns the numeric position of the first occurrence of *needle* in the *haystack* string. Unlike the strrpos(), this function can take a full string as the *needle* parameter and the entire string will be used.
- The parameter haystack is the string to search in
- If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.
- The optional *offset* parameter allows you to specify which character in *haystack* to start searching. The position returned is still relative to the beginning of *haystack* .

Example:

```
<?php
// We can search for the character, ignoring anything before the offset
$newstring = 'abcdef abcdef';
$pos = strpos($newstring, 'a'); // $pos = 0
$pos = strpos($newstring, 'a', 1); // $pos = 7, not 0
?>
```

Preview from Notesale.co.uk
Page 123 of 261

❖ **strrpos**

- Find position of last occurrence of a char in a string

Syntax:

int **strrpos** (string $haystack , string $needle [, int $offset])

- Returns the numeric position of the last occurrence of *needle* in the *haystack* string. Note that the needle in this case can only be a single character in PHP 4. If a string is passed as the needle, then only the first character of that string will be used.
- If *needle* is not found, returns **FALSE**.

Example

```
$nstr='abcdef abcdef';
$pos=strrpos($nstr,'a');        //$pos=7
```

```
// Use of the count parameter is available as of PHP 5.0.0
$str = str_replace("ll", "", "good golly miss molly!", $count);
echo $count; // 2
```

❖ strrev

Syntax:

string **strrev** (string $string)

- Returns *string* , reversed.
- **Parameters:** *string* :The string to be reversed.
- **Return Values:** Returns the reversed string.

Example

```
<?php
echo strrev("Hello world!"); // outputs "!dlrow olleH"
?>
```

❖ echo

- Output one or more strings

Syntax:

void **echo** (string $arg1 [, string $...])

- Outputs all parameters.
- **echo()** is not actually a function (it is a language construct), so you are not required to use parentheses with it.
- **echo()** (unlike some other language constructs) does not behave like a function, so it cannot always be used in the context of a function.
- Additionally, if you want to pass more than one parameter to **echo()**, the parameters must not be enclosed within parentheses.

Example

```
<?php
echo "Hello World";

echo "This spans
multiple lines. The newlines will be
output as well";

echo "This spans\nmultiple lines. The newlines will be\noutput as well.";

echo "Escaping characters is done \"Like this\".";
```

Preview from Notesale.co.uk
Page 126 of 261

Syntax

mixed **max** (array $values)
mixed **max** (mixed $value1 , mixed $value2 [, mixed $value3...])

- **max()** returns the numerically highest of the parameter values.
- If the first and only parameter is an array, **max()** returns the highest value in that array.
- If at least two parameters are provided, **max()** returns the biggest of these values.
- PHP will evaluate a non-numeric string as *0* if compared to integer, but still return the string if it's seen as the numerically highest value. If multiple arguments evaluate to *0*, **max()** will return a numeric *0* if given, else the alphabetical highest string value will be returned.

Example

```
<?php
echo max(1, 3, 5, 6, 7);  // 7
echo max(array(2, 4, 5)); // 5

echo max(0, 'hello');     // 0
echo max('hello', 0);     // hello
echo max(-1, 'hello');    // hello

// With multiple arrays, max compares from left to right
// so in our example: 2 == 2, but 4 < 5
$val = max(array(2, 4, 8), array(2, 5, 7)); // array(2, 5, 7)

// If both an array and non-array are given, the array
// is always returned as it's seen as the largest
$val = max('string', array(2, 5, 7), 42);   // array(2, 5, 7)
?>
```

Preview from Notesale.co.uk
Page 131 of 261

❖ pow

pow — Exponential expression

Syntax

number **pow** (number $base , number $exp)

- Returns *base* raised to the power of *exp* .
- If the power cannot be computed **FALSE** will be returned instead.

Examples

```
<?php
```

⌘12) Date and time function

The PHP date() function is used to format a time or a date.

The PHP date() function formats a timestamp to a more readable date and time.

PHP Date - What is a Timestamp?

A timestamp is the number of seconds since January 1, 1970 at 00:00:00 GMT. This is also known as the Unix Timestamp.

PHP Date - Format the Date

The first parameter in the date() function specifies how to format the date/time. It uses letters to represent date and time formats. Here are some of the letters that can be used:

- d - The day of the month (01-31)
- m - The current month, as a number (01-12)
- Y - The current year in four digits

Other characters, like"/", ".", or "-" can also be inserted between the letters to add additional formatting:

Preview from Notesale.co.uk
Page 133 of 261

```
<?php
echo date("Y/m/d");
echo "<br />";
echo date("Y.m.d");
echo "<br />";
echo date("Y-m-d");
?>
```

The output of the code above could be something like this:

```
2006/07/11
2006.07.11
2006-07-11
```

❖ date — Format a local time/date

syntax:

string date (string $format [, int $timestamp])

- Returns a string formatted according to the given format string using the given integer *timestamp* or the current time if no timestamp is given. In other words, *timestamp* is optional and defaults to the value of time().
- The valid range of a timestamp is typically from fri, 13 dec 1901 20:45:54 GMT to true,19 jan 2038 03:14:07 GMT(these are the dates that corruspond to the minimum and maximum values for a 32-bit signed integer.)

The following characters are recognized in the *format* parameter string

***format* character**	**Description**	**Example returned values**
Day	---	---
D	Day of the month, 2 digits with leading zeros	*01* to *31*
D	A textual representation of a day, three letters	*Mon* through *Sun*
J	Day of the month without leading zeros	*1* to *31*
l (lowercase 'L')	A full textual representation of the day of the week	*Sunday* through *Saturday*
N	ISO-8601 numeric representation of the day of the week (added in PHP 5.1.0)	*1* (for Monday) through *7* (for Sunday)
S	English ordinal suffix for the day of the month, 2 characters	*st*, *nd*, *rd* or *th*. Works well with *j*
W	Numeric representation of the day of the week	*0* (for Sunday) through *6* (for Saturday)
Z	The day of the year (starting from 0)	*0* through *365*
Week	---	---
W	ISO-8601 week number of	Example: *42* (the

Preview from Notesale.co.uk
Page 134 of 261

The following characters are recognized in the *format* parameter string		
***format* character**	**Description**	**Example returned values**
A	Lowercase Ante meridiem and Post meridiem	*am* or *pm*
A	Uppercase Ante meridiem and Post meridiem	*AM* or *PM*
B	Swatch Internet time	*000* through *999*
G	12-hour format of an hour without leading zeros	*1* through *12*
G	24-hour format of an hour without leading zeros	*0* through *23*
H	12-hour format of an hour with leading zeros	*01* through *12*
H	24-hour format of an hour with leading zeros	*00* through *23*
I	Minutes with leading zeros	*00* to *59*
S	Seconds, with leading zeros	*00* through *59*
U	Microseconds (added in PHP 5.2.2)	Example: *54321*
Timezone	---	---
E	Timezone identifier (added in PHP 5.1.0)	Examples: *UTC*, *GMT*, *Atlantic/Azores*
I (capital i)	Whether or not the date is in daylight saving time	*1* if Daylight Saving Time, *0* otherwise.
O	Difference to Greenwich time (GMT) in hours	Example: *+0200*
P	Difference to Greenwich time (GMT) with colon between hours and minutes (added in PHP 5.1.3)	Example: *+02:00*
T	Timezone abbreviation	Examples: *EST*, *MDT* ...
Z	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east	*-43200* through *50400*

Preview from Notesale.co.uk
Page 136 of 261

```
Tue, 24 Jan 2006 14:41:22 CET
1975-10-03T00:00:00+0100
Result with gmdate():
Tuesday
Tuesday 24th of January 2006 01:41:22 PM
Oct 3,1975 was on a Thursday
Tue, 24 Jan 2006 13:41:22 GMT
1975-10-02T23:00:00+0000
```

The output of the code above could be something like this:

```
Tomorrow is 2006/07/12
```

❖ getdate — Get date/time information

syntax:

array getdate ([int $timestamp])

- Returns an associative array containing the date information of the *timestamp*, or the current local time if no *timestamp* is given.
- Key elements of the returned associative array

Preview from Notesale.co.uk
Page 138 of 261

Key	Description	Example returned values
"seconds"	Numeric representation of seconds	*0* to *59*
"minutes"	Numeric representation of minutes	*0* to *59*
"hours"	Numeric representation of hours	*0* to *23*
"mday"	Numeric representation of the day of the month	*1* to *31*
"wday"	Numeric representation of the day of the week	*0* (for Sunday) through *6* (for Saturday)
"mon"	Numeric representation of a month	*1* through *12*
"year"	A full numeric representation of a year, 4	Examples: *1999* or *2003*

syntax:
bool checkdate (int $month , int $day , int $year)

- returns true if the given date is valid otherwise false.Checks the validity of the date formed by the arguments.
- A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given *month* . Leap *year* s are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Examples

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Preview from Notesale.co.uk
Page 140 of 261

The above example will output:

```
bool(true)
bool(false)
```

❖ time — Return current Unix timestamp

syntax:
int time (void)

- Returns the current time measured in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).

Examples

```
<?php
$nextWeek = time() + (7 * 24 * 60 * 60);
                // 7 days; 24 hours; 60 mins; 60secs
```

year

The number of the year, may be a two or four digit value, with values between 0-69 mapping to 2000-2069 and 70-100 to 1970-2000. On systems where time_t is a 32bit signed integer, as most common today, the valid range for *year* is somewhere between 1901 and 2038. However, before PHP 5.1.0 this range was limited from 1970 to 2038 on some systems (e.g. Windows).

is_dst

This parameter can be set to 1 if the time is during daylight savings time (DST), 0 if it is not, or -1 (the default) if it is unknown whether the time is within daylight savings time or not. If it's unknown, PHP tries to figure it out itself. This can cause unexpected (but not incorrect) results. Some times are invalid if DST is enabled on the system PHP is running on or *is_dst* is set to 1. If DST is enabled in e.g. 2:00, all times between 2:00 and 3:00 are invalid and mktime() returns an undefined (usually negative) value. Some systems (e.g. Solaris 8) enable DST at midnight so time 0:30 of the day when DST is enabled is evaluated as 23:30 of the previous day.

Preview from Notesale.co.uk
Page 142 of 261

Note: As of PHP 5.1.0, this parameter became deprecated. As a result, the new timezone handling features should be used instead.

Return Values

mktime() returns the Unix timestamp of the arguments given. If the arguments are invalid, the function returns FALSE (before PHP 5.1 it returned *-1*).

Examples

```
<?php
echo date("M-d-Y", mktime(0, 0, 0, 12, 32, 1997));
echo date("M-d-Y", mktime(0, 0, 0, 13, 1, 1997));
echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 1998));
echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 98));
?>
```

- If *needle* is a string, the comparison is done in a case-sensitive manner.

Parameters

needle

The searched value.

haystack

The array.

strict

If the third parameter *strict* is set to TRUE then the in_array() function will also check the types of the *needle* in the *haystack* .

Return Values:

Returns TRUE if *needle* is found in the array, FALSE otherwise.

Example

```
<?php
$os = array("Mac", "NT", "Irix", "Linux");
if (in_array("Irix", $os)) {
    echo "Got Irix";
}
if (in_array("mac", $os)) {
    echo "Got mac";
}
?>
```

Preview from Notesale.co.uk
Page 146 of 261

The second condition fails because in_array() is case-sensitive, so the program above will display:

```
Got Irix
```

Example

```
<?php
$a = array('1.10', 12.4, 1.13);

if (in_array('12.4', $a, true)) {
    echo "'12.4' found with strict check\n";
}
```

Return Values:

Returns the value of the first array element, or FALSE if the array is empty.

Example

```
<?php

$array = array('step one', 'step two', 'step three', 'step four');

// by default, the pointer is on the first element
echo current($array) . "<br />\n"; // "step one"

// skip two steps
next($array);
next($array);
echo current($array) . "<br />\n"; // "step three"

// reset pointer, start again on step one
reset($array);
echo current($array) . "<br />\n"; // "step one"
end

?>
```

Preview from Notesale.co.uk
Page 149 of 261

❖ **end — Set the internal pointer of an array to its last element**

syntax:
mixed end (array &$array)

- end() advances *array* 's internal pointer to the last element, and returns its value.

Parameters

array

The array.

Return Values:

Returns the value of the last element or FALSE for empty array.

❖ rename — Renames a file or directory

syntax:

bool rename (string $oldname , string $newname [, resource $context])

- Attempts to rename *oldname* to *newname* .

Parameters

oldname

The old name. The wrapper used in *oldname* *must* match the wrapper used in *newname* .

newname

The new name.

context

Context support was added with PHP 5.0.0. For a description of contexts, refer to Stream Functions.

Return Values

Returns TRUE on success or FALSE on failure.

Example

```
<?php
rename("/tmp/tmp_file.txt", "/home/user/login/docs/my_file.txt");
?>
```

Preview from Notesale.co.uk
Page 179 of 261

```
<?php
$email = firstname.lastname@aaa.bbb.com;
$regexp = "/^[^0-9][A-z0-9_]+([.][A-z0-9_]+)*[@][A-z0-
9_]+([.][A-z0-9_]+)*[.][A-z]{2,4}$/";

if (preg_match($regexp, $email)) {
   echo "Email address is valid.";
} else {
   echo "Email address is <u>not</u> valid.";
}
?>
```

This regular expression checks for the number at the beginning and also checks for multiple periods in the user name and domain name in the email address. Let's try to investigate this regular expression yourself.

For the speed reasons, the *preg_match()* function matches only the first pattern it finds in a string. This means it is very quick to check whether a pattern exists in a string. An alternative function, *preg_match_all()*, matches a pattern against a string as many times as the pattern allows, and returns the number of times it matched.

Preview from Notesale.co.uk
Page 190 of 261

Replacing Patterns

In the above examples, we have searched for patterns in a string, leaving the search string untouched. The *preg_replace()* function looks for substrings that match a pattern and then replaces them with new text. *preg_replace()* takes three basic parameters and an additional one. These parameters are, in order, a regular expression, the text with which to replace a found pattern, the string to modify, and the last optional argument which specifies how many matches will be replaced.

preg_replace(pattern, replacement, subject [, limit])

The function returns the changed string if a match was found or an unchanged copy of the original string otherwise. In the following example we search for the copyright phrase and replace the year with the current.

Now let's see a detailed pattern syntax reference:

Regular expression (pattern)	Match (subject)	Not match (subject)	Comment
world	Hello world	Hello Jim	Match if the pattern is present anywhere in the subject
^world	world class	Hello world	Match if the pattern is present at the beginning of the subject
world$	Hello world	world class	Match if the pattern is present at the end of the subject
world/i	This WoRLd	Hello Jim	Makes a search in case insensitive mode
^world$	world	Hello world	The string contains only the "world"
world*	worl, world, worlddd	wor	There is 0 or more "d" after "worl"
world+	world, worlddd	worl	There is at least 1 "d" after "worl"
world?	worl, world, worly	wor, wory	There is 0 or 1 "d" after "worl"
world{1}	world	worly	There is 1 "d" after "worl"
world{1,}	world, worlddd	worly	There is 1 ore more "d" after "worl"
world{2,3}	worldd, worlddd	world	There are 2 or 3 "d" after "worl"
wo(rld)*	wo, world, worldold	wa	There is 0 or more "rld" after "wo"
earth|world	earth, world	sun	The string contains the "earth" or the "world"
w.rld	world, wwrld	wrld	Any character in place of the dot.

Preview from Notesale.co.uk
Page 193 of 261

```
// Print a cookie
echo $_COOKIE["user"];

// A way to view all cookies
print_r($_COOKIE);
?>
```

In the following example we use the isset() function to find out if a cookie has been set:

```
<html>
<body>

<?php
if (isset($_COOKIE["user"]))
  echo "Welcome " . $_COOKIE["user"] . "!<br />";
else
  echo "Welcome guest!<br />";
?>

</body>
</html>
```

Preview from Notesale.co.uk
Page 196 of 261

How to Delete a Cookie?

When deleting a cookie you should assure that the expiration date is in the past.

Delete example:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time()-3600);
?>
```

What if a Browser Does NOT Support Cookies?

If your application deals with browsers that do not support cookies, you will have to use other methods to pass information from one page to another in your application. One method is to

Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

Note: The session_start() function must appear BEFORE the <html> tag:

```
<?php session_start(); ?>
<html>
<body>
</body>
</html>
```

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>

<html>
<body>
<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>
</body>
</html>
```

Output:

Pageviews=1

Preview from Notesale.co.uk
Page 199 of 261

Example

In the following example we create a database called "my_db":

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
if (mysql_query("CREATE DATABASE my_db",$con))
  {
  echo "Database created";
  }
else
  {
  echo "Error creating database: " . mysql_error();
  }
mysql_close($con);
?>
```

Preview from Notesale.co.uk
Page 208 of 261

Create a Table

The CREATE TABLE statement is used to create a database table in MySQL.

Syntax

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
.....
);
```

We must add the CREATE TABLE statement to the mysql_query() function to execute the command.

```
</tr>";
while($row = mysql_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['FirstName'] . "</td>";
  echo "<td>" . $row['LastName'] . "</td>";
  echo "</tr>";
  }
echo "</table>";
mysql_close($con);
?>
```

The **output** of the code above will be:

Firstname	Lastname
Glenn	Quagmire
Peter	Griffin

Preview from Notesale.co.uk
Page 216 of 261

PHP MySQL The Where Clause

To select only data that matches a specified criteria, add a WHERE clause to the SELECT statement.

The WHERE clause

To select only data that matches a specific criteria, add a WHERE clause to the SELECT statement.

Syntax

```
SELECT column FROM table
WHERE column operator value
```

The following operators can be used with the WHERE clause:

Operator	Description
=	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

introduced a new keyword called "**extends**".

While performing **Inheritance** operation Access Specifiers specify the level of access that the outside world (other objects and functions) have on the data members / member functions of the class. During Inheritance operation the derived class can access the parent class attributes having protected/public access specifier. To access private data of parent class from derived class you will have to call public/protected method of the parent class which will access the private variable and return the data to the derived class. In **PHP5** only single inheritance is allowed. i.e. You can inherit only one class.

Example 1 - Parent Keyword in Inheritance Error

```
< ?
class parent     //Generates an error
{
    private $firstname;
    private $lastname;
}

class children extends parent
{
    function __construct()
    {
        echo $this->firstname;
        echo $this->lastname;
    }
}
$a = new children();
?>
```

Preview from Notesale.co.uk
Page 230 of 261

Output: Generate error in PHP5 as parent is keyword in PHP5.
Explanation: Parent is the Keyword in PHP5 and it cannot be used in PHP5 for declaring classname.

Example 2 - Basic Inheritance Call

```
< ?
class parent1
{
    protected $firstname = 11;
    protected $lastname = 23;
}

class children extends parent1
{
```

```
    function __construct()
    {
        echo $this->firstname;
        echo $this->lastname;
    }
}
$a = new children();
?>
```

Your email: 

Output: 1123
Explanation: Parent1 class have extending its functionality to the children class. Now the children class can access the $firstname and $lastname attributes.

Example 3 - Accessing Private Data Member through Inheritance

Preview from Notesale.co.uk
Page 231 of 261

```
< ?
class parent1
{
      private $firstname = "hitesh";
      protected $lastname = 23;

      protected function getData()
      {
            return $this->firstname;
      }
}

class children extends parent1
{
      function __construct()
      {
         echo $this->getData();
      }
}

$a = new children();
?>
```

Output: Hitesh

Serialize Objects:-

- XML is a W3C Recommendation

The Difference Between XML and HTML

XML is not a replacement for HTML.

XML and HTML were designed with different goals:

- XML was designed to transport and store data, with focus on what data is.
- HTML was designed to display data, with focus on how data looks.

HTML is about displaying information, while XML is about carrying information.

XML Does not DO Anything

Maybe it is a little hard to understand, but XML does not DO anything. XML was created to structure, store, and transport information.

The following example is a note to Tove from Jani, stored as XML:

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The note above is quite self descriptive. It has sender and receiver information, it also has a heading and a message body.

But still, this XML document does not DO anything. It is just pure information wrapped in tags. Someone must write a piece of software to send, receive or display it.

XML is Just Plain Text

XML is nothing special. It is just plain text. Software that can handle plain text can also handle XML.

However, XML-aware applications can handle the XML tags specially. The functional meaning of the tags depends on the nature of the application.

Preview from Notesale.co.uk
Page 239 of 261

With XML You Invent Your Own Tags

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

That is because the XML language has no predefined tags.

The tags used in HTML (and the structure of HTML) are predefined. HTML documents can only use tags defined in the HTML standard (like <p>, <h1>, etc.).

XML allows the author to define his own tags and his own document structure.

XML is Not a Replacement for HTML

XML is a complement to HTML.

It is important to understand that XML is not a replacement for HTML. In most web applications, XML is used to transport data, while HTML is used to format and display the data.

My best description of XML is this:

XML is a software- and hardware-independent tool for carrying information.

Preview from Notesale.co.uk
Page 240 of 261

XML is a W3C Recommendation

XML became a W3C Recommendation 10. February 1998.

To read more about the XML activities at W3C.

XML is Everywhere

We have been participating in XML development since its creation. It has been amazing to see how quickly the XML standard has developed, and how quickly a large number of software vendors have adopted the standard.

XML is now as important for the Web as HTML was to the foundation of the Web.

Description

SimpleXMLElement

__construct (string *$data* [, int *$options* [, bool *$data_is_url* [, string *$ns* [, bool *$is_prefix*]]]])

Creates a new **SimpleXMLElement** object.

Parameters

data

A well-formed XML string or the path or URL to an XML document if *data_is_url* is **TRUE**.

options

Optionally used to specify additional Libxml parameters.

data_is_url

By default, *data_is_url* is **FALSE**. Use **TRUE** to specify that *data* is a path or URL to an XML document instead of string data.

ns

is_prefix

Preview from Notesale.co.uk
Page 249 of 261

Return Values

Returns a SimpleXMLElement object representing *data* .

Errors/Exceptions

Produces an **E_WARNING** error message for each error found in the XML data and throws an exception if errors were detected.

Examples

Example #1 Create a SimpleXMLElement object

```
<?php

include 'example.php';

$sxe = new SimpleXMLElement($xmlstr);
echo $sxe->movie[0]->title;
```

```
?>
```

5) SimpleXMLElement::getName — Gets the name of the XML element

Description

SimpleXMLElement

string **getName** (void)

Gets the name of the XML element.

Return Values

The *getName* method returns as a string the name of the XML tag referenced by the SimpleXMLElement object.

Examples

Example #1 Get XML element names

```
<?php
$sxe = new SimpleXMLElement($xmlstr);
echo $sxe->getName() . "\n";
foreach ($sxe->children() as $child)
{
    echo $child->getName() . "\n";
}
?>
```

Preview from Notesale.co.uk
Page 250 of 261

6) simplexml_load_file — Interprets an XML file into an object

Description

object **simplexml_load_file** (string *$filename* [, string *$class_name*= "SimpleXMLElement" [, int *$options*= 0 [, string *$ns* [, bool *$is_prefix*= false]]]])

Convert the well-formed XML document in the given file to an object.

```
</html>
```

As you can see it is just a simple HTML form with a drop down box called "customers".

The below the form will be used as a placeholder for info retrieved from the web server.

When the user selects data, a function called "showUser()" is executed. The execution of the function is triggered by the "onchange" event. In other words: Each time the user change the value in the drop down box, the function showUser() is called.

Example explained - The JavaScript code

This is the JavaScript code stored in the file "selectuser.js":

```
var xmlhttp;

function showUser(str)
{
xmlhttp=GetXmlHttpObject();
if (xmlhttp==null)
  {
  alert ("Browser does not support HTTP Request");
  return
  }
var url="getuser.php";
url=url+"?q="+str;
url=url+"&sid="+Math.random();
xmlhttp.onreadystatechange=stateChanged;
xmlhttp.open("GET",url,true);
xmlhttp.send(null);
}

function stateChanged()
{
if (xmlhttp.readyState==4)
{
document.getElementById("txtHint").innerHTML=xmlhttp.re
sponseText;
}
}

function GetXmlHttpObject()
{
```

Preview from Notesale.co.uk
Page 259 of 261

```
if (window.XMLHttpRequest)
  {
  // code for IE7+, Firefox, Chrome, Opera, Safari
  return new XMLHttpRequest();
  }
if (window.ActiveXObject)
  {
  // code for IE6, IE5
  return new ActiveXObject("Microsoft.XMLHTTP");
  }
return null;
}
```

The stateChanged() and GetXmlHttpObject functions are the same as in the PHP AJAX Suggest chapter, you can go to there for an explanation of those.

The showUser() Function

When a person in the drop-down box is selected, the showUser() function executes the following:

1. Calls the GetXmlHttpObject() function to create an XMLHTTP object
2. Defines an URL (filename) to send to the server
3. Adds a parameter (q) to the URL with the content of the drop-down box
4. Adds a random number to prevent the server from using a cached file
5. Each time the readyState property changes, the stateChanged() function will be executed
6. Opens the XMLHTTP object with the given URL
7. Sends an HTTP request to the server

Preview from Notesale.co.uk
Page 260 of 261

Example explained - The PHP Page

The PHP page called by the JavaScript, is called "getuser.php".

The PHP script runs an SQL query against a MySQL database, and returns the result as HTML:

```
<?php
$q=$_GET["q"];

$con = mysql_connect('localhost', 'peter', 'abc123');
if (!$con)
  {
```