Unit-IV

XML & PHP

Syllabus

UNIT 04

XML : Introduction to XML, uses of XML, simple XML, XML key components, DTD and Schemas, Using XML with application. Transforming XML using XSL and XSLT

PHP: Introduction and basic syntax of PHP, decision and looping with examples, PHP and HTML, Arrays, Functions, Browser control and detection, string, Form processing, Files, Advance Features: Cookies and Sessions, Object Oriented Programming with PHP

*e***X***tensible* **M***arkup* **L***anguage* (XML)

Introduction to XML

XML is a software- and hardware-independent tool for storing and transporting data.

What is XML?

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

The Difference Between XML and HTML

XML and HTML were designed with different goals:

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

How Can XML be Used?

- XML is used in many aspects of web development.
- XML is often used to separate data from presentation.

XML is Often a Complement to HTML

- In many HTML applications, XML is used to store or transport data, while HTML is used to format and display the same data.

How Can XML be Used?

XML Separates Data from HTML

- When displaying data in HTML, you should not have to edit the HTML file when the data changes.
- With XML, the data can be stored in separate XML files.
- With a few lines of JavaScript code, you can read an XML file and update the data content of any HTML page.

What is an XML Element?

- An XML element is everything from (including) the element's start tag to (including) the element's end tag.

```
<price>29.99</price>
```

- An element can contain:
- text
- attributes
- other elements
- or a mix of the above

XML Elements

```xml
<bookstore>
  <book category="children">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML Elements

In the previous example :

- <title>, <author>, <year>, and <price> have **text content** because they contain text (like 29.99).
- <bookstore> and <book> have **element contents**, because they contain elements.
- <book> has an **attribute** (category="children").

XML Elements

Empty XML Elements

- An element with no content is said to be empty.
- In XML, you can indicate an empty element like this:

```
<element></element>
```

- You can also use a so called self-closing tag:

```
<element />
```

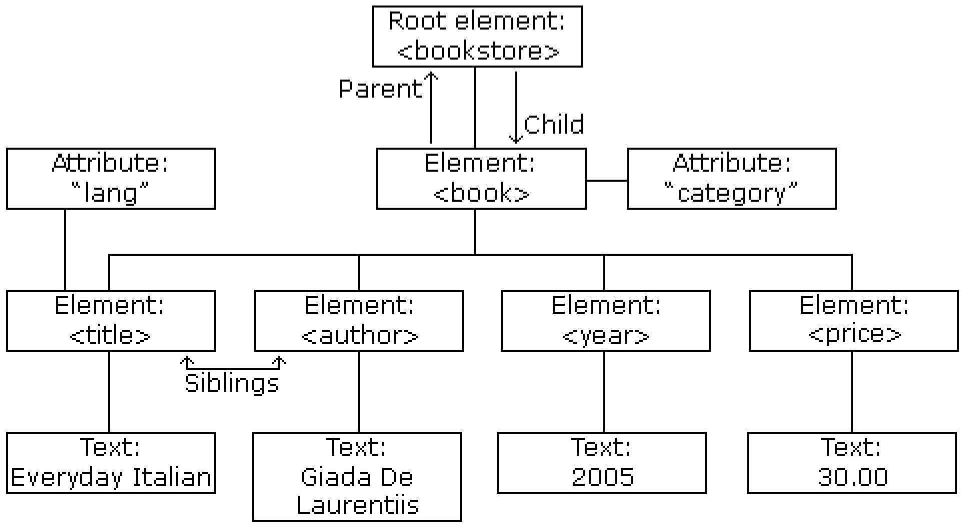
The two forms produce identical results in XML software (Readers, Parsers, Browsers).

Empty elements can have attributes

XML Tree

XML documents form a tree structure that starts at "the root" and branches to "the leaves".

The XML Tree Structure



An Example XML Document

- The image above represents books in this XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML Naming Rules

XML elements must follow these naming rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

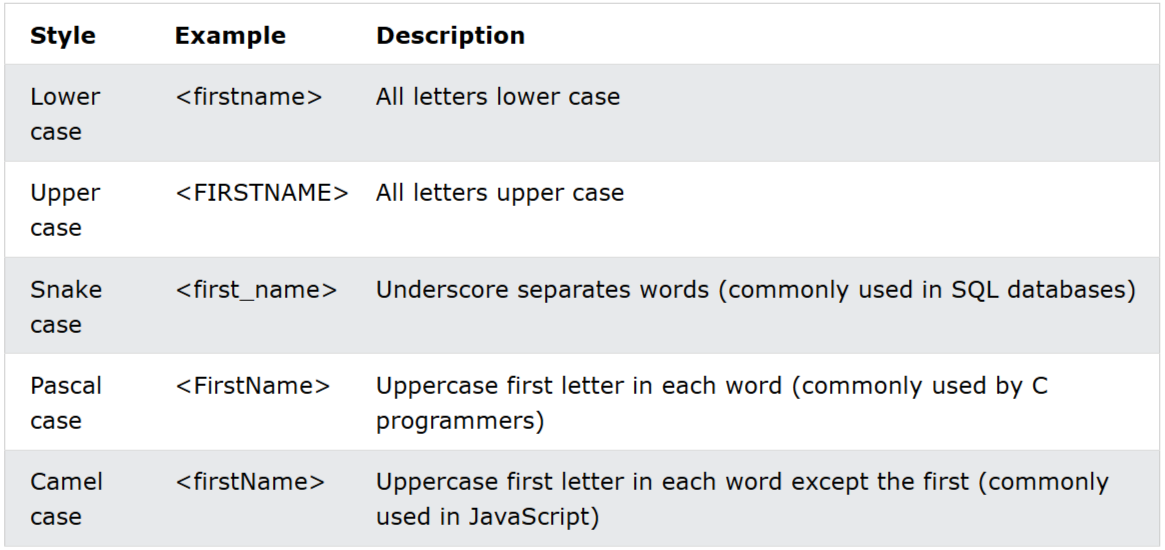
Any name can be used, no words are reserved (except xml).

Best Naming Practices

- Create descriptive names, like this: <person>, <firstname>, <lastname>.
- Create short and simple names, like this: <book_title> not like this: <the_title_of_the_book>.
- Avoid "-". If you name something "first-name", some software may think you want to subtract "name" from "first".
- Avoid ".". If you name something "first.name", some software may think that "name" is a property of the object "first".
- Avoid ":". Colons are reserved for namespaces (more later).
- Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software doesn't support them!

Naming Conventions

- Some commonly used naming conventions for XML elements:



Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Snake case	<first_name>	Underscore separates words (commonly used in SQL databases)
Pascal case	<FirstName>	Uppercase first letter in each word (commonly used by C programmers)
Camel case	<firstName>	Uppercase first letter in each word except the first (commonly used in JavaScript)

XML Attributes

- XML elements can have attributes, just like HTML.
- Attributes are designed to contain data related to a specific element.

XML Attributes Must be Quoted

- Attribute values must always be quoted. Either single or double quotes can be used.
- For a person's gender, the <person> element can be written like this:

`<person gender="female">` or `<person gender='female'>`

XML Attributes

If the attribute value itself contains double quotes you can use single quotes, like in this example:

```
<gangster name='George "Shotgun" Ziegler'>
```

or you can use character entities:

```
<gangster name="George &quot;Shotgun&quot; Ziegler">
```

XML Elements vs. Attributes

Take a look at these two examples:

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>


<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

- In the first example, gender is an attribute. In the last example, gender is an element. Both examples provide the same information.
- There are no rules about when to use attributes or when to use elements in XML.

My Favourite Way

The following three XML documents contain exactly the same information:

A date attribute is used in the first example:

A <date> element is used in the second example:

```
<note>
  <date>2008-01-10</date>
  <to>Tove</to>
  <from>Jani</from>
</note>
```

My Favourite Way

A <date> element is used in the second example:

```
<note>
  <date>2008-01-10</date>
  <to>Tove</to>
  <from>Jani</from>
</note>
```

My Favourite Way

An expanded <date> element is used in the third example: (THIS IS MY FAVORITE):

-
  ```
  <note>
    <date>
      <year>2008</year>
      <month>01</month>
      <day>10</day>
    </date>
    <to>Tove</to>
    <from>Jani</from>
  </note>
  ```

Avoid XML Attributes?

Some things to consider when using attributes are:

- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable (for future changes)

Don't end up like this:

```xml
<note day="10" month="01" year="2008"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

Displaying XML

- Raw XML files can be viewed in all major browsers.
- Don't expect XML files to be displayed as HTML pages.

-
  ```
  <?xml version="1.0" encoding="UTF-8"?>
  - <note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  ```

- Look at the XML file above in your browser: note.xml
- Most browsers will display an XML document with color-coded elements.
- Often a plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure.
- To view raw XML source, try to select "View Page Source" or "View Source" from the browser menu.

Viewing an Invalid XML File

- If an erroneous XML file is opened, some browsers will report the error, and some will display it, or display it incorrectly.

-
  ```
  <?xml version="1.0" encoding="UTF-8"?>
   <note>
    <to>Tove</to>
    <From>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  ```

Why Does XML Display Like This?

- XML documents do not carry information about how to display the data.
- Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table.
- Without any information about how to display the data, the browsers can just display the XML document as it is.

- If you want to style an XML document, use XSLT.

XSLT Introduction

- XSL (eXtensible Stylesheet Language) is a styling language for XML.
- XSLT stands for XSL Transformations.
- This tutorial will teach you how to use XSLT to transform XML documents into other formats (like transforming XML into HTML).

XML DTD

What is a DTD?

- DTD stands for Document Type Definition.
- A DTD defines the structure and the legal elements and attributes of an XML document.

Valid XML Documents

- A "Valid" XML document is "Well Formed", as well as it conforms to the rules of a DTD:

-
  ```xml
  <?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE note SYSTEM "Note.dtd">
  <note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
  </note>
  ```

DTD

-
  ```
  <?xml version="1.0" encoding="UTF-8"?>

  <!DOCTYPE note [
  <!ENTITY nbsp "&#xA0;">
  <!ENTITY writer "Writer: Donald Duck.">
  <!ENTITY copyright "Copyright: W3Schools.">
  ]>

  <note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
  <footer>&writer;&nbsp;&copyright;</footer>
  </note>
  ```

Using XML with application

Display XML Data in an HTML Table

- This example loops through each <CD> element, and displays the values of the <ARTIST> and the <TITLE> elements in an HTML table:

PHP

Hypertext Preprocessor

PHP Introduction

PHP code is executed on the server.

What You Should Already Know

- Before you continue you should have a basic understanding of the following:
  - HTML
  - CSS
  - JavaScript

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

What's new in PHP 7

- PHP 7 is much faster than the previous popular stable release (PHP 5.6)
- PHP 7 has improved Error Handling
- PHP 7 supports stricter Type Declarations for function arguments
- PHP 7 supports new operators (like the spaceship operator: <=>)
- PHP 8: PHP 8.4 is a major update of the PHP language.
- It contains many new features, such as property hooks, asymmetric visibility, an updated DOM API, performance improvements, bug fixes, and general cleanup.

PHP Installation

Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

- install a web server
- install PHP
- install a database, such as MySQL

PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with <?php and ends with ?>:

```
<?php // PHP code goes here ?>
```

PHP Syntax

- <!DOCTYPE html>
- <html>
- <body>

- <h1>My first PHP page</h1>

- <?php
- echo "Hello World!";
- ?>

- </body>
- </html>

Note: PHP statements end with a semicolon (`;`).

PHP Case Sensitivity

- In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.
- In the example below, all three echo statements below are equal and legal:

Example ECHO is the same as echo:

```
<!DOCTYPE html>
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

Decision and Looping in PHP

Decision and looping with examples in php

- PHP Loops
- Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

- Loops are used to execute the same block of code again and again, as long as a certain condition is true.

- In PHP, we have the following loop types:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

PHP while Loop

```
$i = 1;
while ($i < 6)
{
echo $i; $i++;
}
```

The break Statement

```
$i = 1; while ($i < 6)
 {
if ($i == 3)
break;
echo $i; $i++;
}
```

The continue Statement

```
$i = 0;
while ($i < 6)
{
$i++; if ($i == 3)
continue;
echo $i;
}
```

PHP do while Loop

The `do...while` loop will always execute the block of code at least once, it will then check the condition, and repeat the loop while the specified condition is true.

Print `$i` as long as `$i` is less than 6:

```
$i = 1;
do
{
echo $i; $i++;
}
while ($i < 6);
```

PHP do while Loop

The `do...while` loop will always execute the block of code at least once, it will then check the condition, and repeat the loop while the specified condition is true.

Print `$i` as long as `$i` is less than 6:

```php
$i = 1;
do
{
echo $i; $i++;
}
while ($i < 6);
```

The PHP for Loop

The `for` loop is used when you know how many times the script should run.

Syntax

```
for (expression1, expression2, expression3)
{
// code block
}
```

This is how it works:

•*expression1* is evaluated once

•*expression2* is evaluated before each iteration

•*expression3* is evaluated after each iteration

The PHP for Loop

- Print the numbers from 0 to 10:

```php
for ($x = 0; $x <= 10; $x++)
{
echo "The number is: $x <br>";
}
```

Example Explained

1. The first expression, `$x = 0;`, is evaluated once and sets a counter to 0.

2. The second expression, `$x <= 10;`, is evaluated *before* each iteration, and the code block is only executed if this expression evaluates to true. In this example the expression is true as long as `$x` is less than, or equal to, 10.

3. The third expression, `$x++;`, is evaluated *after* each iteration, and in this example, the expression increases the value of `$x` by one at each iteration.

The PHP for Loop with break;

The break Statement

With the `break` statement we can stop the loop even if the condition is still true:

Example

Stop the loop when `$x` is 3:

```
for ($x = 0; $x <= 10; $x++)
{
if ($x == 3) break;
echo "The number is: $x <br>";
}
```

The PHP for Loop with continue;

The continue Statement

With the `continue` statement we can stop the current iteration, and continue with the next:

Example

Stop the loop when `$x` is 3:

```
for ($x = 0; $x <= 10; $x++)
{
if ($x == 3) continue;
echo "The number is: $x <br>";
}
```

PHP foreach Loop

The `foreach` loop - Loops through a block of code for each element in an array or each property in an object.

The foreach Loop on Arrays

The most common use of the `foreach` loop, is to loop through the items of an array.

Loop through the items of an indexed array:

```php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $x)
{
echo "$x <br>";
}
```

For every loop iteration, the value of the current array element is assigned to the variabe `$x`. The iteration continues until it reaches the last array element.

PHP foreach Loop

Keys and Values

The array above is an indexed array, where the first item has the key 0, the second has the key 1, and so on.

Associative arrays are different, associative arrays use named keys that you assign to them, and when looping through associative arrays, you might want to keep the key as well as the value.

This can be done by specifying both the key and value in the `foreach` definition, like this:

Print both the key and the value from the `$members` array:

```
$members = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach ($members as $x => $y)
{
echo "$x : $y <br>";
}
```

PHP and HTML

PHP and HTML

PHP	HTML
PHP is a server-side programming language.	HTML is a client-side scripting language.
PHP is used in backend development, which interacts with databases to retrieve, store, and modify the information.	HTML is used in frontend development, which organizes the content of the website.
PHP is used to create a dynamic website. The output will depend on the browser.	HTML is used to create a static website. The output of static website remains the same on each time.
PHP can manipulate the data.	It cannot manipulate the data.
PHP code executes on web servers like Apache web server, IIS web server.	HTML code executes on web browsers like Chrome, Internet Explorer, etc.
PHP is scripting language.	HTML is a markup language.
PHP8.4 is the latest version of PHP.	HTML5.2 is the latest version of HTML.
PHP is also easy to learn but not as simple as HTML.	HTML is easy to learn. It can easily learn in a very short time.
PHP files save with .php extension.	HTML files save with .html extension.

Arrays in PHP

PHP Arrays

An array stores multiple values in one single variable:

Example

```
$cars = array("Volvo", "BMW", "Toyota");
```

What is an Array?

An array is a special variable that can hold many values under a single name, and you can access the values by referring to an index number or name.

PHP Arrays

PHP Array Types

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index *(Refer Slide 51 foreach loop)*
- **Associative arrays** - Arrays with named keys *(Refer Slide 52 foreach loop)*
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Arrays

PHP Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

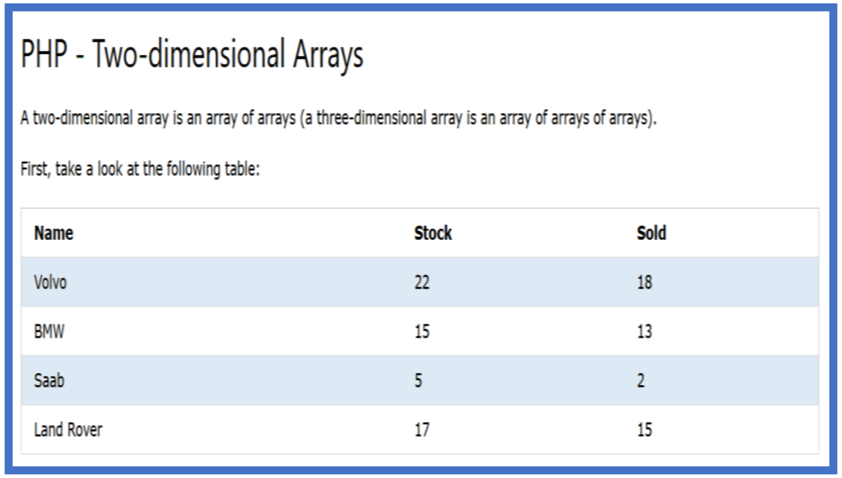
PHP supports multidimensional arrays that are two, three, four, five, or more levels deep.

However, arrays more than three levels deep are hard to manage for most people.

PHP Arrays

The dimension of an array indicates the number of indices you need to select an element.

- For a two-dimensional array you need two indices to select an element
- For a three-dimensional array you need three indices to select an element



PHP - Two-dimensional Arrays

A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

First, take a look at the following table:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

```
$cars = array (
array("Volvo",22,18),
array("BMW",15,13),
array("Saab",5,2),
array("Land Rover",17,15)
);
```

PHP Functions

PHP Functions

The real power of PHP comes from its functions.

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

PHP Built-in Functions

PHP Array Functions

- The array functions allow you to access and manipulate arrays.
- Simple and multi-dimensional arrays are supported.

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

PHP array_change_key_case() Function

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
print_r(array_change_key_case($age,CASE_UPPER));
?>
```

PHP Built-in Functions

PHP Calendar Introduction

The calendar extension contains functions that simplifies converting between different calendar formats.

It is based on the Julian Day Count, which is a count of days starting from January 1st, 4713 B.C.

Note: To convert between calendar formats, you must first convert to Julian Day Count, then to the calendar of your choice.

Note: The Julian Day Count is not the same as the Julian Calendar!

PHP Built-in Functions

PHP Calendar Function

PHP cal_days_in_month() Function

Example

Get the number of days in a month for a specified year and calendar:

```
<?php
$d=cal_days_in_month(CAL_GREGORIAN,02,2024);
echo "There was $d days in October 2024";
?>
```

PHP Built-in Functions

PHP cal_from_jd() Function

Example

Convert a Julian Day Count into a date of in the Gregorian calendar:

Syntax

```
cal_from_jd(jd,calendar);
```

```
<?php
$d=unixtojd(mktime(0,0,0,6,20,2007));
print_r(cal_from_jd($d,CAL_GREGORIAN));
?>
```

Definition and Usage

The cal_from_jd() function converts a Julian Day Count into a date of a specified calendar.

Parameter	Description
jd	Required. Specifies a Julian Day as an integer
calendar	•Required. Specifies the calendar to convert to. Must be one of the following values:CAL_GREGORIAN •CAL_JULIAN •CAL_JEWISH •CAL_FRENCH

PHP Built-in Functions

PHP cal_info() Function

Example

Return information about the Gregorian calendar:

Syntax

```
cal_info(calendar);
```

```
<?php
print_r(cal_info(0));
?>
```

Definition and Usage

The cal_info() function returns information about a specified calendar.

Parameter	Description
calendar	Optional. Specifies a number that indicates which calendar to return information about: 0 = CAL_GREGORIAN 1 = CAL_JULIAN 2 = CAL_JEWISH 3 = CAL_FRENCH **Tip:** If the *calendar* parameter is omitted, cal_info() returns info about all calendars

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

•A function is a block of statements that can be used repeatedly in a program.

•A function will not execute automatically when a page loads.

•A function will be executed by a call to the function.

PHP User Defined Functions

Create a Function

A user-defined function declaration starts with the keyword `function`, followed by the name of the function:

Example

```
function myMessage()
{
echo "Hello world!";
}
```

Call a Function

To call the function, just write its name followed by parentheses `()`:

Example

```
function myMessage()
{
echo "Hello world!";
}
myMessage();
```

PHP Function Arguments

Example

```
function familyName($fname)

{ echo "$fname Refsnes.<br>"; }

familyName("Jani");

familyName("Hege");
```

Browser control
and
Detection

Browser control and detection

PHP get_browser() Function

Example

Look up the browscap.ini file and return the capabilities of the browser:

```
<?php
echo $_SERVER['HTTP_USER_AGENT'];
$browser = get_browser();
print_r($browser);
?>
```

PHP Strings

PHP Strings

Strings

Strings in PHP are surrounded by either double quotation marks, or single quotation marks.

Example

```
echo "Hello";
echo 'Hello';
```

PHP Form Handling/Processing

PHP Form Handling/Processing

The PHP superglobals `$_GET` and `$_POST` are used to collect form-data.

PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

form_processing_welcome.html

```
<html>
<body>
<form action="welcome.php" method="POST">
   Name: <input type="text" name="name"><br>
   E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

form_processing_welcome.php

```
<html>
<body>
Welcome <?php echo $_GET["name"];
?><br>
Your email address is: <?php echo
$_GET["email"]; ?>
</body>
</html>
```

PHP Form Handling/Processing

GET vs. POST

Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)).

This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

PHP Form Handling/Processing

When to use GET?

Information sent from a form with the GET method is **visible to everyone**

The limitation is about 2000 characters

GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is invisible to others.

No limits on the amount of information to send.

Developers prefer POST for sending form data.

PHP File Handling

PHP File Handling

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP Manipulating Files

PHP has several functions for creating, reading, uploading, and editing files.

PHP readfile() Function

The `readfile()` function reads a file and writes it to the output buffer.

Assume we have a text file called "webdictionary.txt", stored on the server, that looks like this:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

PHP File Upload

With PHP, it is easy to upload files to the server.

However, with ease comes danger, so always be careful when allowing file uploads!

Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the `file_uploads` directive, and set it to On:

```
file_uploads = On
```

PHP File Upload

With PHP, it is easy to upload files to the server.

However, with ease comes danger, so always be careful when allowing file uploads!

Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the `file_uploads` directive, and set it to On: `file_uploads = On`

```
file_handling_file_upload.html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-
width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <form action="file_handling_file_upload.php"
enctype="multipart/form-data" method="POST"
name="frm_user_file">
    <input type="file" name="myfile" />
    <input type="submit" name="submit"
value="Upload" />
```

```
file_handling_file_upload.php

<?php
$targetDir = "D:\uploads";
if(is_array($_FILES)) {
if(is_uploaded_file($_FILES['myfile']['t
mp_name'])) {
if(move_uploaded_file($_FILES['myfile'][
'tmp_name'],"$targetDir/".$_FILES['myfil
e']['name'])) {
echo "File uploaded successfully";
}
}
}
?>
```

Advance Features

Cookies and Sessions

PHP Cookies

What is a Cookie?

A cookie is often used to identify a user.

A cookie is a small file that the server embeds on the user's computer.

Each time the same computer requests a page with a browser, it will send the cookie too.

With PHP, you can both create and retrieve cookie values.

PHP Cookies

Create Cookies With PHP

A cookie is created with the `setcookie()` function.

Syntax

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Only the *name* parameter is required. All other parameters are optional.

PHP Cookies

```
<?php
$cookie_name = "admin";
$cookie_value = "vijay";
setcookie($cookie_name, $cookie_value, time() + (86400), "D:\cookies");
// 86400 seconds = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
```

PHP Cookies

To view cookies in Chrome, perform the following steps:

1.Open Chrome settings. ...

2.Choose Inspect to enter the Chrome Developer Tools:

3.Choose the Applications tab. ...

4.Under the Storage tab, select Cookies:

5.Select the website. ...

6.View cookies in Chrome.

PHP Sessions

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the Internet, there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc).

By default, session variables last until the user closes the browser.
So; Session variables hold information about one single user, and are available to all pages in one application.

Start a PHP Session

A session is started with the `session_start()` function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Note: The `session_start()` function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (`session_start()`).

Also notice that all session variable values are stored in the global $_SESSION variable:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Object Oriented Programming with PHP

PHP - What is OOP?

From PHP5, you can also write PHP code in an object-oriented style.

PHP What is OOP?

OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or functions that perform operations on the data,

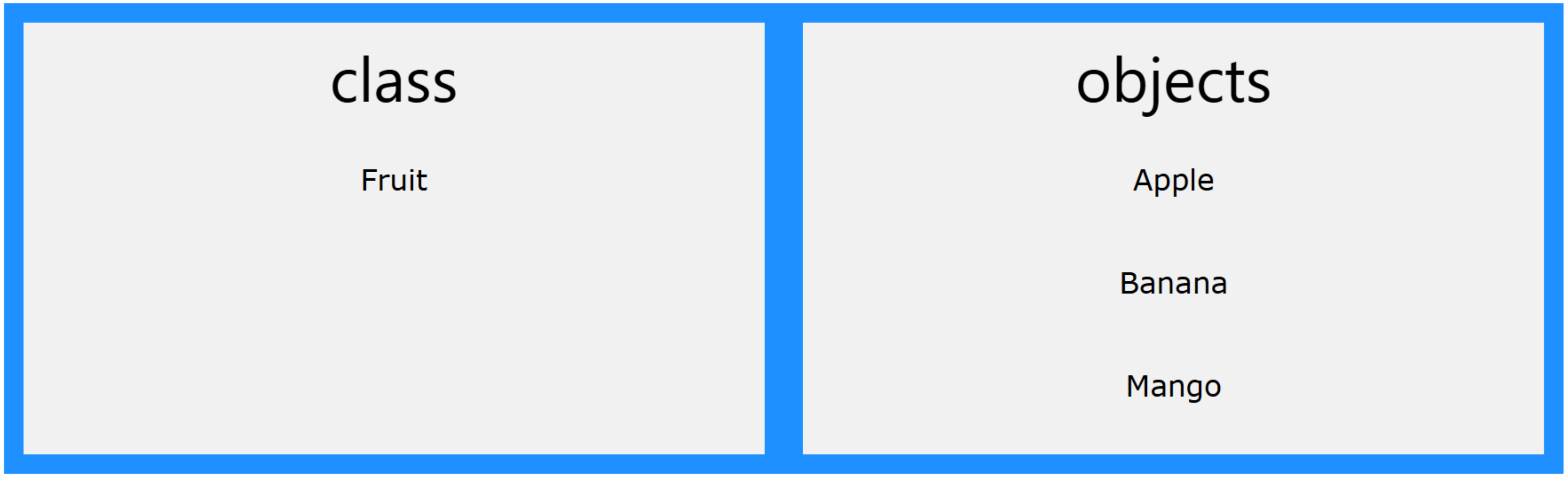
while object-oriented programming is about creating objects that contain both data and functions.

PHP - What is OOP?

PHP - What are Classes and Objects?

Classes and objects are the two main aspects of object-oriented programming.

Look at the following illustration to see the difference between class and objects:



PHP OOP - Classes and Objects

A class is a template for objects, and an object is an instance of class.

Define a Class

A class is defined by using the `class` keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces:

Syntax

```
<?php
class Fruit {
  // code goes here...
}
?>
```

PHP OOP - Classes and Objects

In the example below, we add two more methods to class Fruit, for setting and getting the $color property:

```
<!DOCTYPE html>
<html>
<body>

<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
```

Code Part - 1/2

```
function set_color($color) {
    $this->color = $color;
  }
  function get_color() {
    return $this->color;
  }
}

$apple = new Fruit();
$apple->set_name('Apple');
$apple->set_color('Red');
echo "Name: " . $apple->get_name();
echo "<br>";
echo "Color: " .  $apple->get_color();
?>

</body>
</html>
```

Code Part – 2/2

End of Unit - IV

Thank You