Web Extension: PHP and XML

MODULE 05

XML

- **Xml** (eXtensible Markup Language) is a mark up language.
- XML is designed to store and transport data.
- Xml was released in late 90’s. it was created to provide an easy to use and store self describing data.
- XML is designed to be self-descriptive.
- XML is designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is platform independent and language independent.

Why XML??

- A **mark up language** is a modern system for highlight or underline a document.
- The main benefit of xml is that you can use it to take data from a program like Microsoft SQL, convert it into XML then share that XML with other programs and platforms. You can communicate between two platforms which are generally very difficult.
- The main thing which makes XML truly powerful is its international acceptance. Many corporation use XML interfaces for databases, programming, office application mobile phones and more. It is due to its platform independent feature.

Features and Advantages of XML

1) XML separates data from HTML

2) XML simplifies data sharing

3) XML simplifies data transport

4) XML simplifies Platform change

5) XML increases data availability

6) XML can be used to create new internet languages

Here are some examples:

- **XHTML**
- **WSDL** for describing available web services
- **WAP** and **WML** as markup languages for handheld devices
- **RSS** languages for news feeds
- **RDF** and **OWL** for describing resources and ontology
- **SMIL** for describing multimedia for the web

XML Example

XML documents create a hierarchical structure looks like a tree so it is known as XML Tree that starts at "the root" and branches to "the leaves".

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<note>

 <to>Tove</to>

 <from>Jani</from>

 <heading>Reminder</heading>

 <body>Don't forget me this weekend!</body>

</note>
```

XML

XML documents must contain a **root element.** This element is "the parent" of all other elements.

The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree.

All elements can have sub elements (child elements).

```
<root>
 <child>
  <subchild>.....</subchild>
 </child>
</root>
```

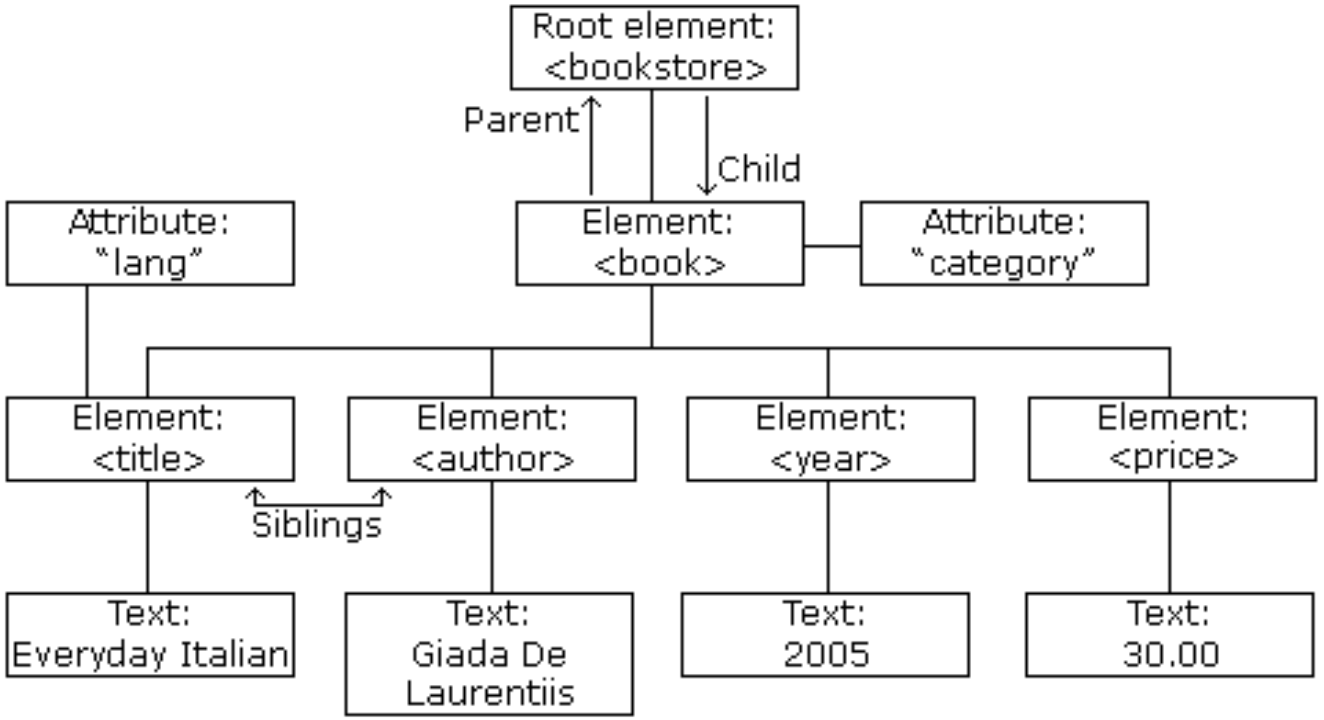
Example of XML: Books

```
<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>
```

he <book> element has 4 children: <title>,< author>, <year> and <price>.

```
  <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 Tree Structure



XML DTD

- An XML document with correct syntax is called "Well Formed".
- An XML document validated against a DTD is both "Well Formed" and "Valid".
- DTD stands for Document Type Definition.
- A DTD defines the structure and the legal elements and attributes of an XML document.

XML DTD

The purpose of a DTD is to define the structure and the legal elements and attributes of an XML document

Note.dtd:

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

The DTD above is interpreted like this:

- !DOCTYPE note - Defines that the root element of the document is note
- !ELEMENT note - Defines that the note element must contain the elements: "to, from, heading, body"
- !ELEMENT to - Defines the to element to be of type "#PCDATA"
- !ELEMENT from - Defines the from element to be of type "#PCDATA"
- !ELEMENT heading - Defines the heading element to be of type "#PCDATA"
- !ELEMENT body - Defines the body element to be of type "#PCDATA"
- #PCDATA means parseable character data.

Use of DTD

When to Use a DTD?

- With a DTD, independent groups of people can agree to use a standard DTD for interchanging data.
- With a DTD, you can verify that the data you receive from the outside world is valid.
- You can also use a DTD to verify your own data.

When NOT to Use a DTD?

- XML does not require a DTD.
- When you are experimenting with XML, or when you are working with small XML files, creating DTDs may be a waste of time.

XML Schema

- An XML Schema describes the structure of an XML document, just like a DTD.
- XML Schema Definition(XSD)
- XML Schema is an XML-based alternative to DTD
- <xs:element name="note">

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

- <xs:element name="note"> defines the element called "note"
- <xs:complexType> the "note" element is a complex type
- <xs:sequence> the complex type is a sequence of elements
- <xs:element name="to" type="xs:string"> the element "to" is of type string (text)
- <xs:element name="from" type="xs:string"> the element "from" is of type string
- <xs:element name="heading" type="xs:string"> the element "heading" is of type string
- <xs:element name="body" type="xs:string"> the element "body" is of type string

XML Schema

XML Schemas are More Powerful than DTD

- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- XML Schemas support namespaces

XML DOM

- The DOM defines a standard for accessing and manipulating documents.
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

The HTML DOM defines a standard way for accessing and manipulating HTML documents. It presents an HTML document as a tree-structure.

The XML DOM defines a standard way for accessing and manipulating XML documents. It presents an XML document as a tree-structure.

The XML DOM

```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>

</bookstore>
```

```
//txt = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
```

This example loads a text string into an XML DOM object, and extracts the info from it with JavaScript:

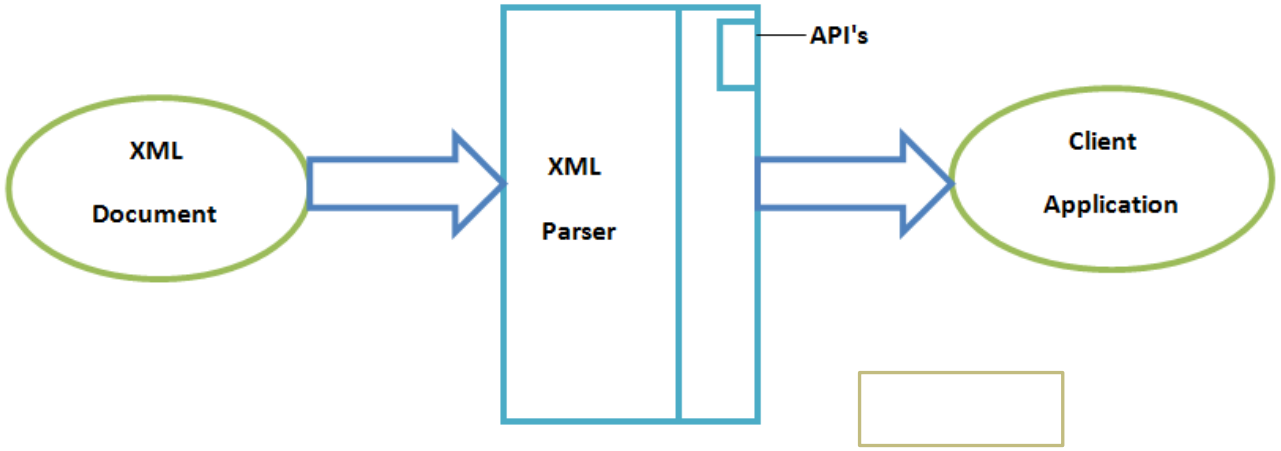
```
<html>
<body>

<p id="demo"></p>

<script>
var text, parser, xmlDoc;

text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>

</body>
</html>
```

XML Parser

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document.

The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.



Types of XML Parsers –DOM ,SAX

DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

Advantages

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

Disadvantages

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

2) It is comparatively slower than other parsers.

SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

Features of SAX Parser

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

Advantages

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

Disadvantages

1) It is event-based so its API is less intuitive.

2) Clients never know the full information because the data is broken into pieces.

Parsing a Text String

```
<html>
<body>

<p id="demo"></p>

<script>
var text, parser, xmlDoc;

text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>

</body>
</html>
```

XSL (Extensible Stylesheet Language)

XSL is a language for expressing style sheets.

An XSL style sheet is, like with CSS, a file that describes how to display an XML document of a given type.

- A transformation language for XML documents: **XSLT**. Originally intended to perform complex styling operations, like the generation of tables of contents and indexes, it is now used as a general purpose XML processing language.
- XSLT is thus widely used for purposes other than XSL, like generating HTML web pages from XML data.
- Advanced styling features, expressed by an XML document type which defines a set of elements called **Formatting Objects**, and attributes (in part borrowed from CSS2 properties and adding more complex ones.

XSLT Elements and Attributes

1. The <xsl:attribute> element is used to add attributes to elements.

Syntax:

```
<xsl:attribute name="attributename" namespace="uri">

</xsl:attribute>
```

Example:

```
<picture>
   <xsl:attribute name="source"/>
</picture>
```

XSL : value-of

The <xsl:value-of> element can be used to select the value of an XML element and add it to the output.

Syntax:

```
<xsl:value-of select="expression" disable-output-escaping="yes|no" />
```

Attribute	Value	Description
select	expression	Required. An XPath expression that specifies which node/attribute to extract the value from.
disable-output-escaping	yes no	Optional. "yes" indicates that special characters (like "<") should be output as is. "no" indicates that special characters (like "<") should be output as "<". Default is "no"

XSL : for-each

The <xsl:for-each> element loops through each node in a specified node set.

Syntax

```
<xsl:for-each select="expression">
  <!-- Content -->
</xsl:for-each>
```

Attribute	Value	Description
select	expression	Required. An XPath expression that specifies which node set to be processed.

XSL :sort

The <xsl:sort> element is used to sort the output.

Syntax

```
<xsl:sort select="expression"
lang="language-code"
data-type="text|number|qname"
order="ascending|descending"
case-order="upper-first|lower-first"/>
```

XSL :if

The <xsl:if> element contains a template that will be applied only if a specified condition is true.

Syntax

```
<xsl:if test="expression">

  <!-- Content: template -->

</xsl:if>
```

Simple xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
</catalog>
```

XSL Style Sheet

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

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

```
▼<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
 ▼<xsl:template match="/">
  ▼<html>
   ▼<body>
      <h2>My CD Collection</h2>
     ▼<table border="1">
      ▼<tr bgcolor="#9acd32">
         <th style="text-align:left">Title</th>
         <th style="text-align:left">Artist</th>
       </tr>
      ▼<xsl:for-each select="catalog/cd">
       ▼<tr>
        ▼<td>
           <xsl:value-of select="title"/>
         </td>
        ▼<td>
           <xsl:value-of select="artist"/>
         </td>
        </tr>
       </xsl:for-each>
     </table>
    </body>
   </html>
  </xsl:template>
</xsl:stylesheet>
```

Link the XSL Style Sheet to the XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers

Introduction to PHP

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side.

PHP is well suited for web development.

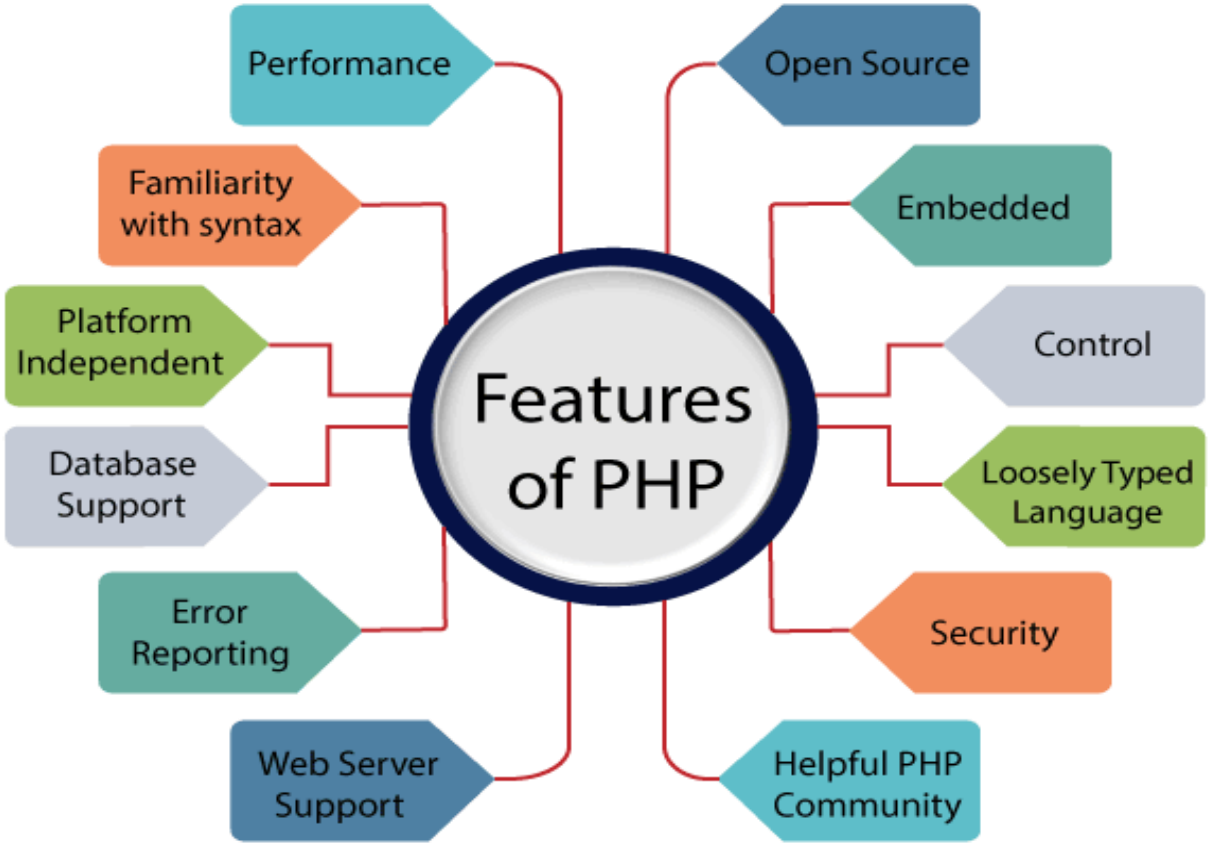
Therefore, it is used to develop web applications (an application that executes on the server and generates the dynamic page.).

PHP was created by **Rasmus Lerdorf in 1994** but appeared in the market in 1995.

PHP

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is simple and easy to learn language.

PHP Features



Install PHP

To install PHP, need to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

- **WAMP** for Windows
- **LAMP** for Linux
- **MAMP** for Mac
- **SAMP** for Solaris
- **FAMP** for FreeBSD
- **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail, etc.

PHP Example

Step 1: Create a simple PHP program like hello world.

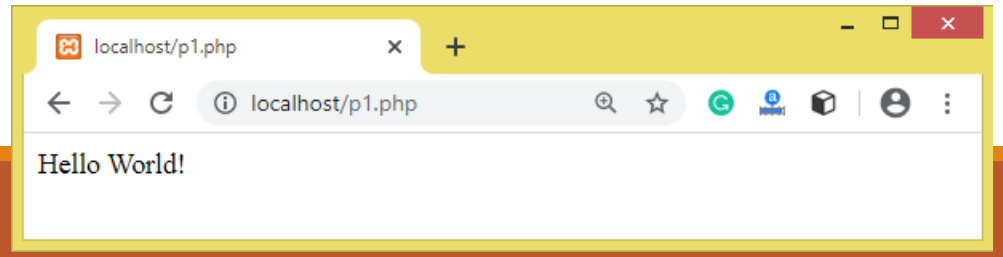
```
<?php
   echo "Hello World!";
?>
```

Step 2: Save the file with **p1.php** name in the htdocs folder, which resides inside the xampp folder.

Step 3: Run the XAMPP server and start the Apache and MySQL.

Step 4: Now, open the web browser and type localhost *http://localhost/hello.php* on your browser window.

Step 5: The output for the above **p1.php** program will be shown as the screenshot below:



PHP Case Sensitivity

In PHP, keyword (e.g., echo, if, else, while), functions, user-defined functions, classes are not case-sensitive. However, all variable names are case-sensitive.

```
<!DOCTYPE>
<html>
  <body>
    <?php
      echo "Hello world using echo </br>";
      ECHO "Hello world using ECHO </br>";
      EcHo "Hello world using EcHo </br>";
    ?>
  </body>
</html>
```

PHP Echo

PHP echo is a language construct, not a function.

Therefore, you don't need to use parenthesis with it.

But if you want to use more than one parameter, it is required to use parenthesis.

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

PHP Variables

In PHP, a variable is declared using a **$ sign** followed by the variable name.

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.
- **Syntax:**
- $variablename=value;

PHP Variables

- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

string is: hello string integer is: 200 float is: 44.6

PHP Data Types

PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types (predefined)
2. Compound Types (user-defined)
3. Special Types

PHP Data Types: Scalar Types

It holds only single value. There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float
4. string

PHP Data Types

PHP Data Types: Compound Types

It can hold multiple values. There are 2 compound data types in PHP.

1. array
2. object

PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

Example

```
<?php
   $dec1 = 34;
   $oct1 = 0243;
   $hexa1 = 0x45;
   echo "Decimal number: " .$dec1. "</br>";
   echo "Octal number: " .$oct1. "</br>";
   echo "HexaDecimal number: " .$hexa1. "</br>";
?>
```

PHP Array

```
1.<?php
2.    $bikes = array ("Royal Enfield", "Yamaha", "KTM");
3.    var_dump($bikes);   //the var_dump() function returns the datatype and values
4.    echo "</br>";
5.    echo "Array Element1: $bikes[0] </br>";
6.    echo "Array Element2: $bikes[1] </br>";
7.    echo "Array Element3: $bikes[2] </br>";
8.?>
```

PHP object

Objects are the instances of user-defined classes that can store both values and functions. They must be explicitly declared.

```
<?php
    class bike {
        function model() {
            $model_name = "Royal Enfield";
            echo "Bike Model: " .$model_name;
          }
    }
    $obj = new bike();
    $obj -> model();
?>
```

PHP Control Structures

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

- if
- if-else
- if-else-if
- nested if
- Switch case

PHP Loops

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

The PHP while Loop

Syntax

```
while (condition is true) {
  code to be executed;
}
```

The PHP do...while Loop

Syntax

```
do {
  code to be executed;
} while (condition is true);
```

The PHP for Loop

Syntax

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

PHP foreach Loop

Syntax

```
foreach ($array as $value) {
  code to be executed;
}
```

PHP Functions

PHP function is a piece of code that can be reused many times.

It can take input as argument list and return value.

There are thousands of built-in functions in PHP.

Syntax

```
1.function functionname(){
2.//code to be executed
3.}
```

PHP Form Handling

Simple HTML form

```
<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>
```

Welcome.php

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

O/p:

```
Welcome XYZ
Your email address is XYZ@example.com
```

PHP MySQL Database

PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.

PHP MySQL Connect Example

```php
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
{
  die('Could not connect: ' . mysqli_error());
}
echo 'Connected successfully';
mysqli_close($conn);
?>
```

PHP MySQL Create Database

```
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
{
  die('Could not connect: ' . mysqli_connect_error());
}
echo 'Connected successfully<br/>';
$sql = 'CREATE Database mydb';
if(mysqli_query( $conn,$sql)){
  echo "Database mydb created successfully.";
}else{
echo "Sorry, database creation failed ".mysqli_error($conn);
}
mysqli_close($conn);
?>
```