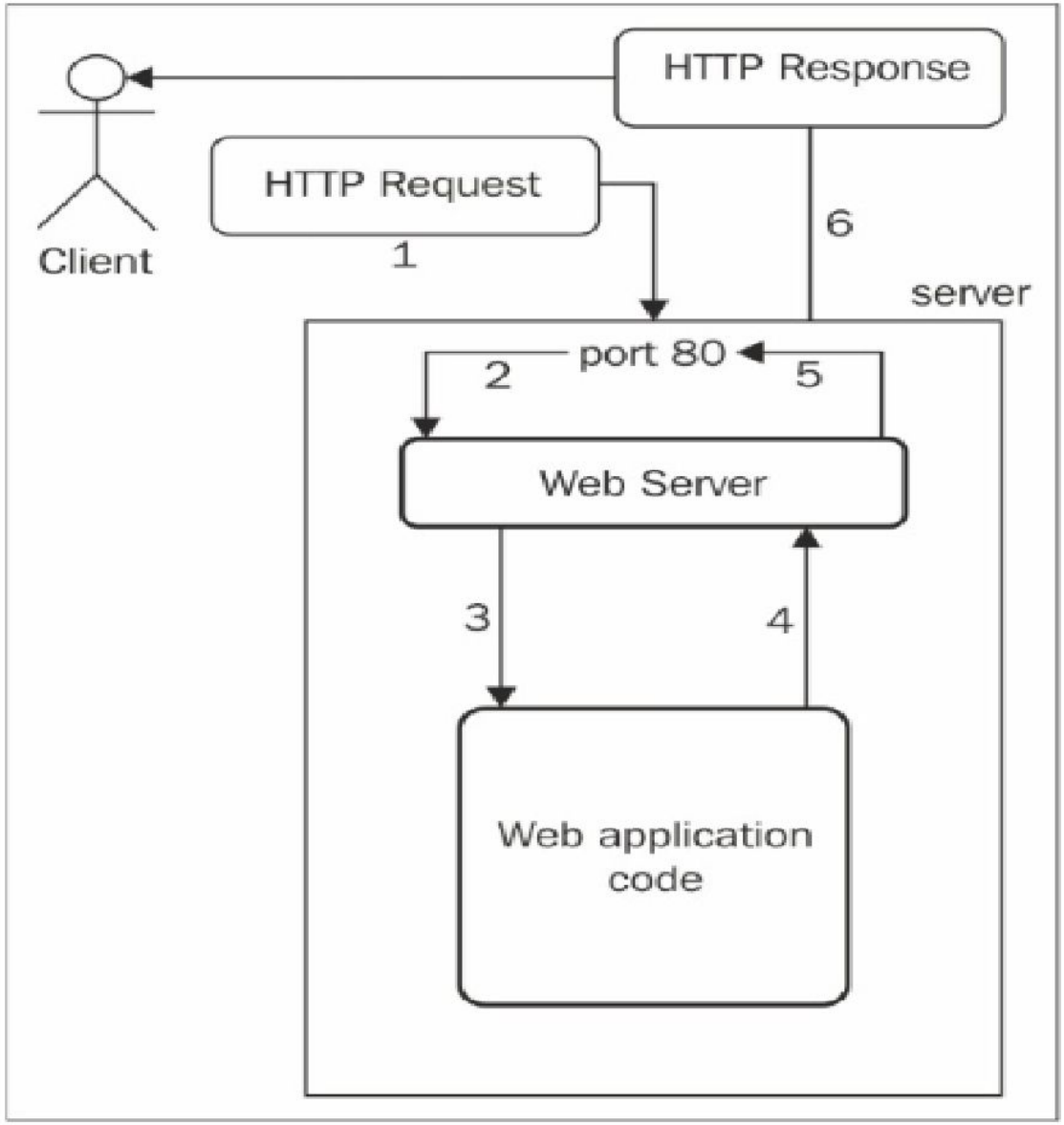
Introduction to PHP

Introduction

- A web page is a single document with content. It contains links that open other web pages with different content.
- A website is the set of web pages that usually live in the same server and are related to each other.
- A web application is just a piece of software that runs on a client, which is usually a browser, and communicates with a server. A server is a remote machine that receives

Basic Architecture

The job of a web server is to route external requests to the correct application so that they can be processed. Once the application returns a response, the web server will send this response to the client.



Request-response flow on the server side

Steps

1. The client, which is a browser, sends a request. This can be of any type—GET or POST—and contain anything as long as it is valid.
2. The server receives the request, which points to a port. If there is a web server listening on this port, the web server will then take control of the situation.
3. The web server decides which web application—usually a file in the filesystem needs to process the request.

   In order to decide, the web server usually considers the path of the URL; for example, http://myserver.com/app1/hi would try to pass the request to the app1 application, wherever it is in the filesystem.

4. The web application, after receiving a request from the web server, generates a response and sends it to the web server.

5. The web server sends the response to the indicated port.

6. The response finally arrives to the client.

What is PHP?

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

→ PHP is a very popular and widely-used open source server-side scripting language to write dynamically generated web pages. PHP was originally created by Rasmus Lerdorf in 1994. It was initially known as Personal Home Page.

→ PHP scripts are executed on the server and the result is sent to the web browser as plain HTML.

→ PHP can be integrated with the number of popular databases, including MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Sybase, and so on.

What is a PHP File?

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

- A PHP script is executed on the server, and the plain HTML result is sent back to the browser
- A PHP script can be placed anywhere in the document.
- PHP script starts with **<?php** and ends with **?>**

```
<?php

// PHP code goes here

?>
```

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
- With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

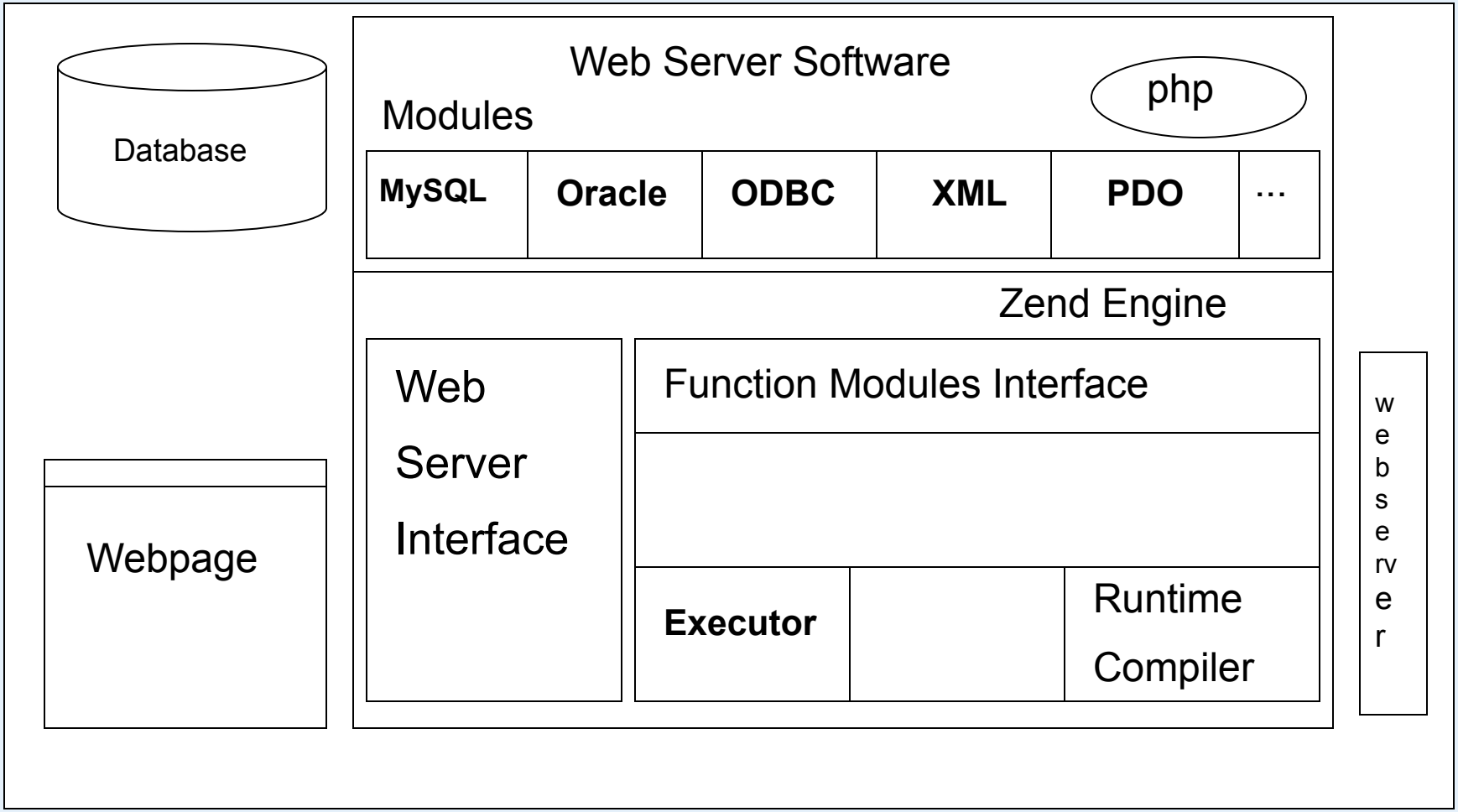
- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently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

Zend & PHP

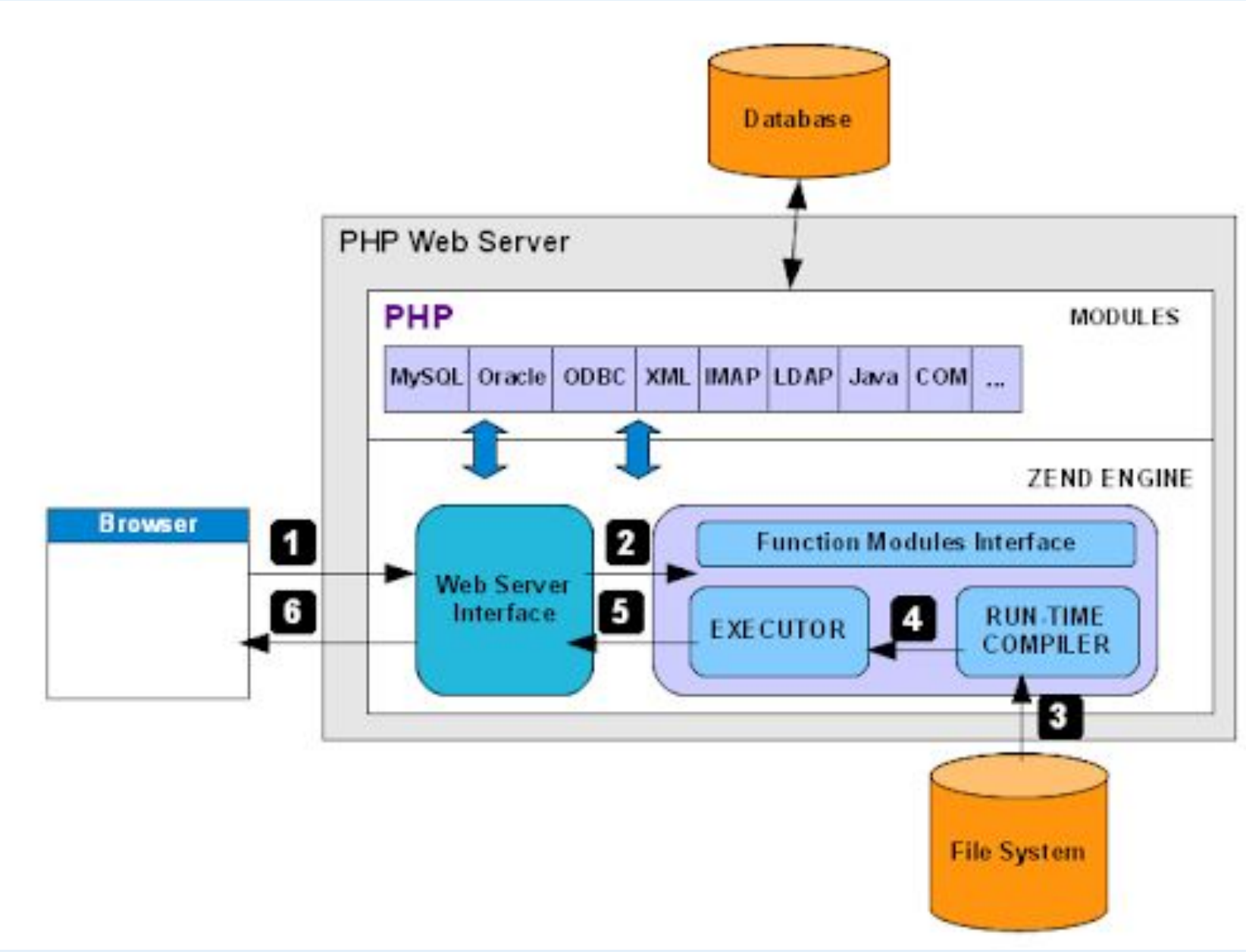
- Zend refers to the language engine - PHP's core.
- The Zend Engine is an open source scripting engine opcode-based: (a Virtual Machine), commonly known for the important role it plays in the web automation language PHP.

- **Zend** Framework (ZF) is an open source, object-oriented web application framework implemented in **PHP**.
- The **Zend Engine** is the open source scripting **engine** that interprets the PHP programming language.
  - It was originally developed by Zeev Suraski and Andi Gutmans while they were students at the Technion - Israel Institute of Technology.
  - They later founded a company called **Zend** Technologies in Ramat Gan, Israel.

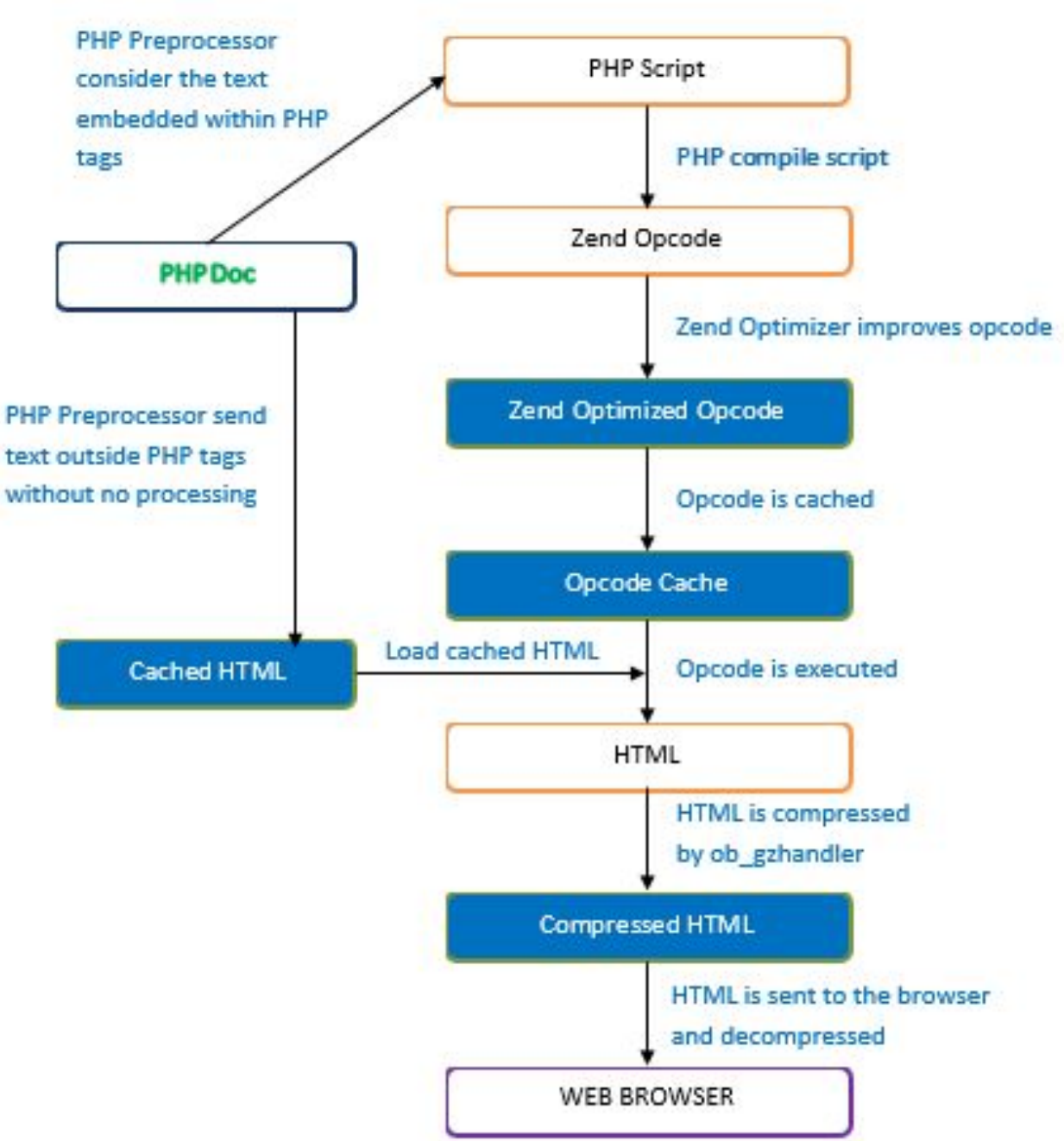
Internal Structure of PHP



Architecture Overview



- **External modules** can be loaded from the disk at script runtime using the function “bool dl (string $library)”. After the script is terminated, the external module is discarded from memory.
- **Built-in modules** are compiled directly into PHP and carried around with every PHP process; their functionality is available to every script that's being run.
- **Memory Management**: Zend gets full control over all memory allocations in fact it determines whether a block is in use, automatically freeing unused blocks and blocks with lost references, and thus prevent memory leaks.
- **Zend Executor**: Zend Engine compiles the PHP Code in the intermediate code *Opcode* which is executed by the Zend Executor which converts it to machine language.


PHP Preprocessor consider the text embedded within PHP tags
PHP Script
PHP compile script
Zend Opcode
PHP Doc
Zend Optimizer improves opcode
Zend Optimized Opcode
PHP Preprocessor send text outside PHP tags without no processing
Opcode is cached
Opcode Cache
Cached HTML
Load cached HTML
Opcode is executed
HTML
HTML is compressed by ob_gzhandler
Compressed HTML
HTML is sent to the browser and decompressed
WEB BROWSER

```
<html>

<head>

<title>Party List</title>

</head>

<body>

<?php
$guest[00]="Irma";
$guest[01]="Salvatore";
$guest[02]="Caterina";
$guest[03]="Simone";
?>
<p> The list of participants to the event is: </p>
<ol>
<?php
Foreach ($aGuest as $Guest) {
Echo "<li>".$aGuest."</li>;
};
?>
</ol>
</body>
</html>
```

Server Pre processing

```
rint "<html>";
Print "<head>";
Print "<title>Party List</title>";
Print "</head>";
Print "<body>";
$guest[00]="Irma";
$guest[01]="Salvatore";
$guest[02]="Caterina";
$guest[03]="Simone";
Print "<p> The list of participants to the event is: </p>";
Print "<ol>";
Foreach ($aGuest as $Guest) {
Echo "<li>".$aGuest."</li>;
};
Print "</ol>";
Print "</body>";
rint "</html>";
```

Zend Engine Processing

```
<html>
<head>
<title>Party List</title>
</head>
<body>
<p> The list of participants to the event is: </p>
<ol>
<li>Irma</li>
<li>Salvatore</li>
<li>Caterina</li>
<li>Simone</li>
</ol>
</body>
</html>
```

How PHP Server Engine works

1. The script is run through a ***lexical analyzer*** to convert the human-readable code into tokens. These tokens are then passed to the parser.
2. The ***parser*** parses, manipulates and optimizes the stream of tokens passed to it from the lexical analyzer and generates an intermediate code called ***opcodes*** (an ordered array of instructions) that runs on the Zend Engine. This two steps which represents the compilation phase are provided by the Run-Time Compiler module.
3. After the intermediate code is generated, it is passed to the Executor. The executor steps through the ***op array***, using a function for each opcode and HTML is generated for the same.
4. This generated HTML is sent to client, if the web browser supports compressed web pages the HTML is encoded using gzip or deflate before sending.
5. This opcode is flushed from memory after execution.

PHP Installation

- To run PHP Script at any web server, three components are required :-
  - An **interpreter** that analyzes PHP code snippets, translates them to something CPU understands, checks the translation for any kind of error & finally passes it to CPU for execution.
  - A **functionality** section of the interpreter which implements functionality of PHP.
  - An **interface** section that talks to the Web Server.

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

PHP Installation

- PHP can be configured on IIS, WAMP and XAMPP servers.
  - **IIS**(Internet Information Server) is a web-server application just like Apache is, except it's made by Microsoft and is Windows only (Apache runs on both Windows and Linux).
  - **XAMPP** is an acronym for **X** (any Operating System), **A**pache (Web server), **M**ySQL Database, **P**HP Language and **P**ERL.
  - **WAMP** is an acronym for **W**indows (OS), **A**pache (web-server), **M**ySQL (database), **P**HP (language).
  - **IIS** is also more geared towards using ASP.NET (vs. PHP) and "SQL Server" (vs. MySQL), though it can use PHP and MySQL too.

XAMPP or WAMP?

- XAMPP is easy to use than WAMP.
- XAMPP is more powerful.
- XAMPP has a control panel from that you can start and stop individual components (such as MySQL, Apache etc.).
- XAMPP is more resource consuming than WAMP because of heavy amount of internal component software like Tomcat , FileZilla FTP server, Webalizer, Mercury Mail etc.
- So if you do not need high features better to go with WAMP.
- XAMPP also has SSL feature which WAMP doesn't.
  - SSL(Secure Sockets Layer (SSL) is a networking protocol that manages server authentication, client authentication and encrypted communication between servers and clients.)

Conclusion: If your applications need to deal with native web apps only, Go for WAMP. If you need advanced features as stated above, go for XAMPP.

Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>.**
- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- PHP statements end with a semicolon (;).

- Let us take an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page.

Example

```
<html>
   <head></head>
   <body>
      <h1>My first PHP page</h1>
      <?php
         echo "Hello World!";
      ?>
   </body>
</html>
```

Comments in PHP

- A comment in PHP code is a line that is not read/executed as part of the program.
- Its only purpose is to be read by someone who is looking at the code.
- It can be written using //, #, /*...*/

- Comments can be used to:
  - Let others understand what you are doing
  - Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did.
  - Comments can remind you of what you were thinking when you wrote the code

Several ways of commenting

```
<html>
<body>
  <?php
   // This is a single-line comment
   # This is also a single-line comment
  /*
  This is a multiple-lines comment block
  that spans over multiple lines
  */

// You can also use comments to leave out parts
  of a code line
  $x = 5 /* + 15 */ + 5;
  echo $x;
  ?>
</body>
</html>
```

Case Sensitivity

- In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.
- E.g.

```
<html>
<body>

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

</body>
</html>
```

Case Sensitivity

- However; all variable names are case-sensitive. E.g.

```
<html>
<body>

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

- Here only the first statement will display the value of the $color variable (this is because $color, $COLOR, and $coLOR are treated as three different variables)

Variables

- In PHP, a variable starts with the $ sign, followed by the name of the variable

```
<?php
  $txt = "Hello world!";
  $x = 5;
  $y = 10.5;
?>
```

- When you assign a text value to a variable, put quotes around the value.
- Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Variables

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- Rules for PHP variables:
  - A variable **starts with the $ sign**, followed by the name of the variable
  - A variable name must **start with a letter or the underscore character**
  - A variable name **cannot start with a number**
  - A variable name **can only contain alpha-numeric characters and underscores** (A-z, 0-9, and _)
  - Variable names **are case-sensitive** ($age and $AGE are two different variables)

Output Variables

- The PHP echo statement is often used to output data to the screen.
- Example 1:

```
<?php
  $txt = “PHP learning";
  echo "I love $txt!";
?>
```

- Example 2:

```
<?php
  $txt = " Welcome to PHP learning ";
  echo "I love " . $txt . "!";
?>
```

For concatenate the message with variable in echo **dot operator is** used.

One more example

- Sum of two variables

```
<?php
  $x = 5;
  $y = 4;
  echo $x + $y;
?>
```

Is a Loosely Typed Language

- We do not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.
- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.
- **Type juggling**

  PHP tries to convert the data type of a variable only when there is a context where the type of variable needed is different. But PHP does not change the value and type of the variable itself. Instead, it will take the value and try to transform it, leaving the variable intact.

- PHP is also known as a dynamically typed language. Explicit type declaration of a variable is neither needed nor supported in PHP.
- Contrary to C, C++ and Java, type of PHP variable is decided by the value assigned to it, and not other way around.
- Further, a variable when assigned value of different type, its type too changes. This approach of PHP to deal with dynamically changing value of variable is called type juggling.

Example

```
<?php
    $a = "1";
    $b = 2;
    var_dump($a + $b); // 3
    var_dump($a . $b); // 12
?>
```

Var_dump () : gives the information about the variable

Example:

```
<?php
   $var="Hello"; // variable is string
   type
   $var=100; //same variable now
   becomes int

   $var1=100;
   $var2="100";
   $var3=$var1+$var2;
   var_dump($var3);

   $var2="100 days";
   $var3=$var1+var2;
   var_dump($var3);
?>
```

Example: Casting

```
<?php

   $var1=100;
   $var2=(boolean)$var1;
   $var3=(string)$var1;
   $var4=(array)$var1;
   $var5=(object)$var1;
   var_dump($var2, $var3, $var4, $var5);

   $var1=100.50;
   $var2=(string)$var1;
   $var3="$var1";
   var_dump($var2, $var3);
?>
```

Testing type of variable

- To test the type of variable use gettype().
- string **gettype** (mixed $var)
- Mixed - $var can be of any type.

· **Example :- datatype**

```
<?php
$testvar;
echo gettype($testvar);//Null
echo "<br>";

$testvar = 15;
echo gettype($testvar);
  //integer
echo "<br>";

$testvar = "ABC";
echo gettype($testvar); //string
$testvar = 15.80;

echo gettype($testvar); // Float
echo "<br>";
$testvar = true;
echo gettype($testvar); //Boolean
echo "<br>";
?>
```

Testing variable for specific type

Function	Description
is_int($var)	Returns true if $var is an integer
is_float($var)	Returns true if $var is a float
is_string($var)	Returns true if $var is a string
is_bool($var)	Returns true if $var is a boolean
is_array($var)	Returns true if $var is an array
is_object($var)	Returns true if $var is an object
is_resource($var)	Returns true if $var is a resource
is_null($var)	Returns true if $var is a null

Changing variable's datatype

- Using settype() :-
  - To use settype pass the name of variable which you want to change the type and new datatype
  - bool **settype** (mixed &$var , string $type)
- Using Type casting :-
  - PHP provides type conversion using cast operator

Operator	Changes To
int or integer	Integer
float, real, double	Floating point
String	String
bool or boolean	Boolean
array	Array
Object	Object

Changing variable's datatype

- PHP function to cast a value

Function	Description
intval($var or value)	Returns value cast to integer
floatval($var or value)	Returns value cast to float
strval($var or value)	Returns value cast to string

Datatypes

- PHP has several different types of variables.
- All holds specific class/type of information.
- PHP has 8 basic data types which are categorized into 3 categories :-
  - Scalar Datatype
  - Compound Datatype
  - Special Datatype

Scalar Data Type

- Scalar data means data that contains only a single value.

Data type	Description
Integer	- Stores whole numbers either +ve or – ve. - If value assigned is out of the range it is converted to float.
Float	- Store floating point numbers as well as higher integers.
String	- Stores sequence of characters terminated by NULL.
Boolean	- It holds true or false. - Internally it holds integer value - 0 for false - rest values for true.

Compound Data types

- Compound data is data that can contain more than one variable.

Data type	Description
Array	- Ordered map - It holds multiple values.
Object	- They have multiple values within it. - They have their own functions for accessing and/ or manipulating its data.

Special Datatype

- PHP supports two special data types which have a specific meaning.

Data types	Description
Resource	Contains a reference to an external resource, such as file or database. It can be used in same manner as we use another variables
Null	Contains null as a value. Explicitly do not contain any value.

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

Global and Local Scope

- **A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function**
- E.g.

```
<?php
  $x = 5; // global scope

  function myTest() {
      // using x inside this function will generate
  an error
      echo "<p>Variable x inside function is:
  $x</p>";
  }
  myTest();

  echo "<p>Variable x outside function is: $x</p>";
?>
```

Cont...

- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function
- E.g.

```
<?php
  function myTest() {
      $x = 5; // local scope
      echo "<p>Variable x inside function is:
  $x</p>";
  }
  myTest();

  // using x outside the function will generate an
  error
  echo "<p>Variable x outside function is: $x</p>";
?>
```

  - You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

The global Keyword

- The global keyword is used to access a global variable from within a function.
- To do this, use the global keyword before the variables (inside the function)
- E.g.
- ```
<?php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```
```

The static Keyword

- When a function is completed/executed, all of its variables are deleted.
- To do this, use the **static** keyword when you first declare the variable.

```
<html> <body>
<?php
        function myTest() {
              static $x = 0;
              echo $x;
              $x++;
        }
        myTest();     echo "<br>";
        myTest();     echo "<br>";
        myTest();     echo "<br>";
        myTest();     echo "<br>";
        myTest();
?>
</body> </html>
```

echo and print Statements

- In PHP there are two basic ways to get output: echo and print.
- We use echo (and print) in almost every example.
- echo and print are more or less the same. They are both used to output data to the screen.
- The differences are small:
  - echo has no return value while print has a return value of 1 so it can be used in expressions.
  - echo can take multiple parameters (although such usage is rare) while print can take one argument.
  - echo is marginally faster than print.

The echo Statement

- The echo statement can be used with or without parentheses: echo or echo().
- Example 1

```
<?php
  echo "<h2>PHP is Fun!</h2>";
  echo "Hello world!<br>";
  echo "I'm about to learn PHP!<br>";
  echo "This ", "string ", "was ", "made ",
  "with multiple parameters.";
?>
```

The print Statement

- **The print statement can be used with or without parentheses: print or print().**
- Example 1

```
<?php
  print "<h2>PHP is Fun!</h2>";
  print "Hello world!<br>";
  print "I'm about to learn PHP!";
?>
```

The print Statement

- **Example 2**

```
<?php
  $txt1 = "Learn PHP";
  $txt2 = " Welcome to PHP learning";
  $x = 5;
  $y = 4;

  print "<h2>$txt1</h2>";
  print "Study PHP at $txt2<br>";
  print $x + $y;
?>
```

Data Types

- PHP supports the following data types:
  - String
  - Integer
  - Float (floating point numbers - also called double)
  - Boolean
  - Array
  - Object
  - NULL
  - Resource

String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

- **Example**

```
<?php
  $x = "Hello world!";
  $y = 'Hello world!';

  echo $x;
  echo "<br>";
  echo $y;
?>
```

Integer

- An integer is a whole number (without decimals).
- range is between -2147483648 and 2147483647 in 32 bit systems,
- and between -9223372036854775808 and 9223372036854775807 in 64 bit systems.
- A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.
- Rules for integers:
  - An integer must have at least one digit (0-9)
  - An integer cannot contain comma or blanks
  - An integer must not have a decimal point
  - An integer can be either positive or negative
  - Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
- **Example**

```
<?php
    $x = 5985;
    var_dump($x);
?>
```

- PHP has the following predefined constants for integers:
  - PHP_INT_MAX - The largest integer supported
  - PHP_INT_MIN - The smallest integer supported
  - PHP_INT_SIZE - The size of an integer in bytes

- PHP has the following functions to check if the type of a variable is integer:
  - is_int()
  - is_integer() - alias of is_int()
  - is_long() - alias of is_int()

Example

```
<?php
   $x = 500;
   $x = $x * 0.1;
   var_dump($x);
?>
```

Float

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- In the following example $x is a float. The PHP var_dump() function returns the data type and value:
- **Example**

```
<?php
    $x = 10.365;
    var_dump($x);
?>
```

  - N.B. The **var_dump()** function is used to display structured information (type and value) about one or more variables.

- The float data type can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits.
- PHP has the following predefined constants for floats (from PHP 7.2):
  - PHP_FLOAT_MAX - The largest representable floating point number
  - PHP_FLOAT_MIN - The smallest representable positive floating point number
  - PHP_FLOAT_DIG - The number of decimal digits that can be rounded into a float and back without precision loss
  - PHP_FLOAT_EPSILON - The smallest representable positive number x, so that x + 1.0 != 1.0
- PHP has the following functions to check if the type of a variable is float:
  - is_float()
  - is_double() - alias of is_float()

Infinite Numbers

- A numeric value that is larger than PHP_FLOAT_MAX is considered infinite.

- PHP has the following functions to check if a numeric value is finite or infinite:
  - is_finite()
  - is_infinite()

Example:

```
<?php
        $x = 1.9e555;
        var_dump($x);
?>
```

NaN - Not a Number

- NaN stands for Not a Number.
- NaN is used for impossible mathematical operations.

- PHP has the following functions to check if a value is not a number:
  - is_nan()

Example:

```
<?php
    $x = acos(8);
    var_dump($x);
?>
```

- The PHP is_numeric() function can be used to find whether a variable is numeric.
- The function returns true if the variable is a number or a numeric string, false otherwise.

Example:

```
<?php
   $x = 5985;
   var_dump(is_numeric($x));

   $x = "5985";
   var_dump(is_numeric($x));

   $x = "59.85" + 100;
   var_dump(is_numeric($x));

   $x = "Hello";
   var_dump(is_numeric($x));
?>
```

Boolean

- A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

- Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

Array

- An array stores multiple values in one single variable.
- E.g.

```
<?php
    $cars = array("Maruti", "Tata",
      "Hyundai", “KIA”, “Mahindra”);
    var_dump($cars);
?>
```

- Here $cars is an array. The PHP var_dump() function returns the data type and value

Object

- An object is a data type which stores data and information on how to process that data.
- In PHP, an object must be explicitly declared.
- First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods. E.g.

```
<?php
class Car {
    function Car() {
      $this->model = "VW";
    }
}
// create an object
$herbie = new Car();
// show object properties
echo $herbie->model;
?>
```

NULL Value

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- **Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.
- Variables can also be emptied by setting the value to NULL

```
<?php
    $x = "Hello world!";
    $x = null;
    var_dump($x);
?>
```

Strings

- A string is a sequence of characters, like "Hello world!".
- **Get The Length of a String**
  - **Strlen():** function returns the length of a string (number of characters).
  - The example below returns the length of the string "Hello world!":

```
<?php
  echo strlen("Hello world!"); // outputs 12
?>
```

Cont...

- **Count The Number of Words in a String**
- **str_word_count()** function counts the number of words in a string:

Example

```
<?php

   echo str_word_count("Hello world!");
   // outputs 2

?>
```

Cont...

- **Reverse a String**
- **strrev()** function reverses a string. E.g.

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

Cont...

Search For a Specific Text Within a String

- **strpos()** function searches for a specific text within a string.
- If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

- *Syntax :* strpos(string,find,start)
- String : specifies the string to search
- Find : Specifies the string to find
- Start : Specifies where to begin the search. If *start* is a negative number, it counts from the end of the string.
- E.g.

```
<?php
    echo strpos("Hello world!", "world"); //
     outputs 6
    ?>
```

Cont...

- **Replace Text Within a String**
- **str_replace()** function replaces some characters with some other characters in a string.
- E.g.

```
<?php
   echo str_replace("world", "Dolly",
    "Hello world!");
   // outputs Hello Dolly!
?>
```

Getting Substring from string

Substr () function extracts the string contained between the positions specified by parameters, with the first character being at position 0.

syntax : `substr(`*`string`*`,`*`start`*`,`*`length`*`)`

string : source string

start : starting position of the string

length: length of return string

```
<?php
   echo substr("Hello world", 3, 5);
   // outputs lo wo
?>
```

Constants

- A constant is an identifier (name) for a simple value whose value cannot be changed during the script.
- A valid constant name starts with a letter or underscore **(no $ sign before the constant name).**
- Unlike variables, constants are automatically global across the entire script. The syntax is as follows:
    - define(*name*, *value*, *case-insensitive*) where,
        - *name*: Specifies the name of the constant
        - *value*: Specifies the value of the constant
        - *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

Cont...

- **Example 1**

```
<?php
define("GREETING", "Welcome to PHP
 World!");
echo GREETING;
?>
```

- **Example 2**

```
<?php
 define("GREETING", "Welcome to PHP
 World!", true);
 echo greeting;
?>
```

Constants are Global

- **Constants are automatically global** and can be used across the entire script.
- **Example**

```
<?php
 function myTest() {
 define(“message", "Welcome to world of
 PHP!");
  echo message;
 }
 myTest();
 echo message;
?>
```

Differences between constants and variables

- There is no need to write a dollar sign ($) before a constant, where as in Variable one has to write a dollar sign.
- Constants cannot be defined by simple assignment, they may only be defined using the define() function.
- Constants may be defined and accessed anywhere without regard to variable scoping rules.
- Once the Constants have been set, may not be redefined or undefined.