-PHP Introduction

PHP is the contracted of Hypertext Preprocessor and earlier it was contracted as Personal Home Page.

PHP is a server-side scripting language designed specifically for web development. PHP can be easily implanted in HTML files and HTML codes can also be written in a PHP file.

PHP codes are executed on the server whereas HTML codes are directly rendered on the browser.

It is open-source which means it is free to download and use.

It can be merged with many databases such as Oracle, Microsoft SQL Server, MySQL, PostgreSQL, Sybase, Informix.

Websites like www.facebook.com, www.yahoo.com

Set PHP development environment in windows

There are basically two ways to set up PHP on a local machine which are:

- Using all in one package (XAMPP & WAMPP).
- Manually install all the required packages (MySQL, PHP & Apache) and configure them.

Downloading XAMPP: You can download the XAMPP software from the official website here

PHP syntax
The script starts with <?php and ends with ?>. These tags are also called ‘Canonical PHP tags’.

```
<?php
        // Here echo command is used to print
        echo "Hello, world!";
?>
```

PHP | Variables

Variables in a program are used to store some values or data that can be used later in a program.

The variables are also like containers that store character values, numeric values, memory addresses, and strings.

```
$a =54;
$b=”aniket”
```

PHP Data Types

- String- “Hello” / ‘hii’
- Integer- 23, 56
- Float- 23.0 ,56.5
- Boolean - true/false
- Array- $a=array(“hello”,”hii”,”aniket”,87657)
- Object-
- NULL
- **Resource --** A common example of using the resource data type is a database call.

PHP var_dump() function

The PHP var_dump() function returns the data type and value.
Ex- 14

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

$y = 10.365;
var_dump($y);

$a = true;
$b = false;

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

$c = null;
var_dump($c);
?>
```

PHP Casting Strings and float

```php
<?php
// Cast float to int
$x = 23465.768;
$x = (int)$x;
echo $x;

echo "<br>";

// Cast string to int
$y = "23465.768";
$y = (int)$y;
echo $y;
?>
```

PHP Math

min() and max() Functions

```php
echo(min(0, 150, 30, 20, -8, -200)); // returns -200
```

```php
echo(max(0, 150, 30, 20, -8, -200)); // returns 150
```

sqrt() Function

```php
echo(sqrt(64)); // returns 8
```

Random Numbers

`rand()` function generates a random number:

```php
echo(rand());
echo(rand(10, 100));
```

PHP Operators

Operators are used to perform operations on variables and values.

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

PHP Arithmetic Operators

Operator Name Example Result + Addition $x + $y Sum of $x and $y

- Subtraction $x - $y Difference of $x and $y * Multiplication $x * $y

Product of $x and $y / Division $x / $y Quotient of $x and $y

% Modulus $x % $y Remainder of $x divided by $y

** Exponentiation $x ** $y Result of raising $x to the $y power

PHP Assignment Operators

Assignment Same as... Description

x = y x = y The left operand gets set to the value of the expression on the right

x += y x = x + y Addition

x -= y x = x - y Subtraction

x *= y x = x * y Multiplication

x /= y x = x / y Division

x %= y x = x % y Modulus

PHP Comparison Operators

Operator Name Example Result

== Equal $x == $y Returns true if $x is equal to $y

=== Identical $x === $y Returns true if $x is equal to $y, and they are of the same

type and data type

!= Not equal $x != $y Returns true if $x is not equal to $y **not match**

<> Not equal $x <> $y Returns true if $x is not equal to $y **not match**

!== Not identical $x !== $y Returns true if $x is not equal to $y, or they are not of the same type

> Greater than $x > $y Returns true if $x is greater than $y

< Less than $x < $y Returns true if $x is less than $y

than or equal to $y

>= Greater than or equal to

$x <= $y Returns true if $x is less than or equal to $y

<= Less than or equal to

$x >= $y Returns true if $x is greater

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator Name Example Result

and And $x and $y True if both $x and $y are true z

or Or $x or $y True if either $x or $y is true
xor Xor $x xor $y True if either $x or $y is true, but not both

&& And $x && $y True if both $x and $y are true

|| Or $x || $y True if either $x or $y is true

! Not !$x True if $x is not true

PHP Conditional Statements

Conditional statements are used to perform different actions based on different conditions. ,

In PHP we have the following conditional statements:

- `if` statement - executes some code if one condition is true
- `if...else` statement - executes some code if a condition is true and another code if that condition is false
- `if...elseif...else` statement - executes different codes for more than two conditions
- `switch` statement - selects one of many blocks of code to be executed

PHP - The if...elseif...else Statement

```
1. <?php
2.     $marks=69;
3.     if ($marks<33){
4.         echo "fail";
5.     }
6.     else if ($marks>=34 && $marks<50) {
7.         echo "D grade";
8.     }
```

```
9.      else if ($marks>=50 && $marks<65) {
10.       echo "C grade";
11.    }
12.    else if ($marks>=65 && $marks<80) {
13.       echo "B grade";
14.    }
15.    else if ($marks>=80 && $marks<90) {
16.       echo "A grade";
17.    }
18.    else if ($marks>=90 && $marks<100) {
19.       echo "A+ grade";
20.    }
21.   else {
22.       echo "Invalid input";
23.    }
24. ?>
```

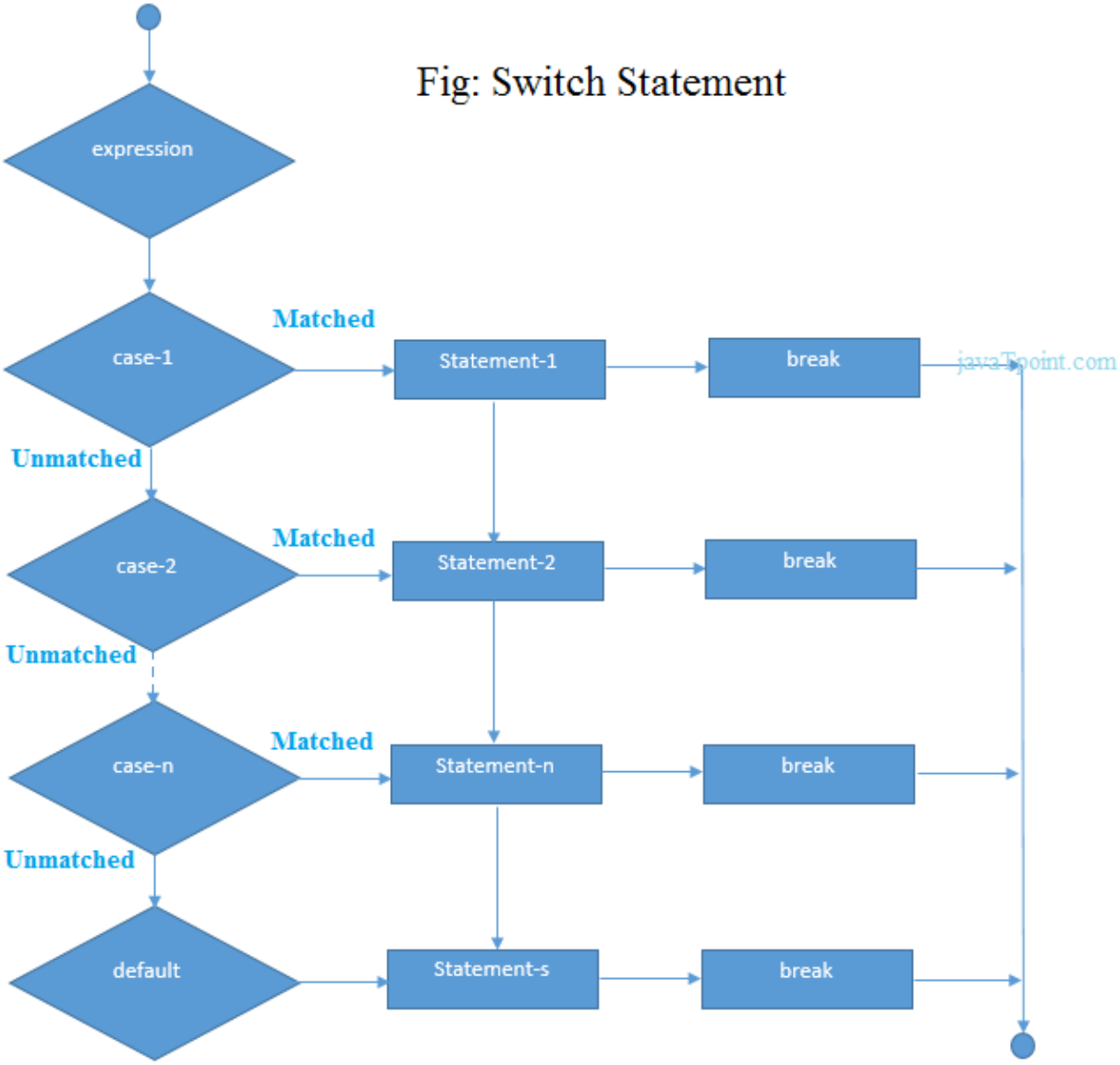
The PHP switch Statement

The `switch` statement is used to perform different actions based on different conditions.

1. The **default** is an optional statement. Even it is not important, that default must always be the last statement.
2. There can be only one **default** in a switch statement. More than one default may lead to a **Fatal** error.

3. The **break** statement is optional to use in switch. If break is not used, all the statements will execute after finding matched case value

Fig: Switch Statement




```php
<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite ";
    break;
```

```
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP Loops

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

The purpose of the loop is to repeat the same code a number of times.

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

while

```
<?php
$i=1;
while($i <= 3) {
   echo $i;
   $i++;
}
?>
```

Do..while it will check the condition after processing an iteration and before go to next iteration. So, in *do..while* there is an assured guaranty for the execution of the first iteration.

```
<?php
$i=1;
do{
   echo $i;
   $i++;
}while($i <= 3);
?>
```

For loop

PHP for loop can be used to traverse set of code for the specified number of times.It should be used if the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code.It allows users to put all the loop related statements in one place. See in the syntax given below:

Structure

```
for(initialization; conditional statement; increments (or) decrements) {
```

```
   //set of expressions to be execute recursively
}
```

Example

```
<?php
for($i=1;$i <= 3;$i++) {
   echo $i;
}
?>
```

Foreach

The *foreach* loop is used to iterate the array variables to process the array keys and values. The foreach loop is used to traverse the array elements. It works only on array and object. It will issue an error if you try to use it with the variables of different datatype.

The foreach loop works on elements basis rather than index. It provides an easiest way to iterate the elements of an array.

In foreach loop, we don't need to increment the value.

Syntax

```
For each ($array as $value) {

   code to be executed;

}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Example

```
<?php

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {

  echo "$value <br>";
}

?>
```

PHP Break

The `break` statement can be used to jump out of a loop and switch The **break** keyword immediately ends the execution of the loop or switch structure. It breaks the current flow of the program at the specified condition and program control resumes at the next statements outside the loop.statement. `<?php`

```
for ($x = 0; $x < 10; $x++) {

  if ($x == 4) {

    break;

  }
```

```
  echo "The number is: $x <br>";
}
?>
```

PHP Continue

The `continue` statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.The continue statement is used within looping and switch control structure when you immediately jump to the next iteration.

The continue statement can be used with all types of loops such as - for, while, do-while, and foreach loop. The continue statement allows the user to skip the execution of the code for the specified condition.

```
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
  }
  echo "The number is:". $x . "<br>";
}
?>
```

What Is an Array?

array is a data structure which allows you to store multiple elements in a single variable. elements are stored as key-value pairs

```
<?php
$array_fruits = array('Apple', 'Orange', 'Watermelon', 'Mango');
?>
```

How to Initialize an Array

```
<?php
$array = array();
?>
```

add elements to an array.

```
<?php
$array = [];
$array[0] = 'One';
$array[1] = 'Two';
$array[2] =’Three';
```

```
print_r($array);
?>
```

Access Array Elements

Example1

```
<?php
$array = ['One', 'Two', 'Three'];

// get the first element of the $array array
echo $array[0];
echo "<br>";

// get the second element of the $array array
echo $array[1];
echo "<br>";


// get the third element of the $array array
echo $array[2];
echo "<br>";
?>
```

Example2

```
<?php
$array = ['One', 'Two', 'Three'];

foreach ($array as $var) {
  echo $var;
  echo '<br>';
```

```
}
?>
```

Example3

```
<?php
$array = ['One', 'Two', 'Three'];
$array_length = count($array);

for ($i = 0; $i < $array_length; ++$i) {
  echo $array[$i];
  echo '<br>';
}
?>
```

Types of Arrays in PHP

1-Indexed Arrays

The numeric index is assigned automatically when you don’t specify it explicitly.

```
<?php
$array = ['One', 'Two', 'Three'];

print_r($array);
$array_length = count($array);

for ($i = 0; $i < $array_length; ++$i) {
  echo $array[$i];
  echo '<br>';
```

```
}
?>
```

2-Associative Arrays

An associate array is similar to an indexed array, but you can use string values for array keys.

```
<?php
$employee = [
  'name' => 'John',
  'email' => 'john@example.com',
  'phone' => '1234567890',
];

// get the value of employee name
echo $employee['name'];

// get all values
foreach ($employee as $key => $value) {
  echo $key . ':' . $value;
  echo '<br>';
}
?>
```

Multidimensional Arrays

store arrays as elements within other arrays—this is a multidimensional array.

```
<?php
$employee = [
  'name' => 'John',
  'email' => 'john@example.com',
  'phone' => '1234567890',
  'hobbies' => ['Football', 'Tennis'],
  'profiles' => ['facebook' => 'johnfb', 'twitter' => 'johntw']
];

// access hobbies
echo $employee['hobbies'][0];
// Football

echo $employee['hobbies'][1];
// Tennis
// access profiles
echo $employee['profiles']['facebook'];
// johnfb

echo $employee['profiles']['twitter'];
// johntw
?>
```

PHP File Handling

A file is a container in computer storage devices used for storing data.

Types of Files

Text files

Binary files

Binary files are mostly the .bin files in your computer.

File Operations

- Creating a new file
- Opening an existing file
- Closing a file
- Reading from and writing information to a file

readfile() Function

The PHP code to read the file . The `readfile()` function returns the number of bytes read on success:

```
<?php
echo readfile("hello.txt");
?>
```

fopen()

A better method to open files is with the `fopen()` function. This function gives you more options than the `readfile()` function.

fread()

fread() function reads from an open file.

fread(name_of_file , size_of_file)

fclose()

The `fclose()` function is used to close an open file.

```
<?php
$myfile = fopen("hello.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("hello.txt"));
fclose($myfile);
```

```
?>
```

fgets()

The `fgets()` function is used to read a single line from a file. fgetc()

The `fgetc()` function is used to read a single character from a file.

```
<?php
$file = fopen("abc.txt", "r") or die("Unable to open file!");
echo fgets($file);
fclose($file);
?>
```

Modes Description
r Open a file for read-only.

w Open a file for write-only. Erases the contents of the file or creates a new file if it doesn't exist.

a Open a file for write-only. The existing data in file is preserved. Creates a new file if the file doesn't exist

x Creates a new file for write only. Returns FALSE and an error if file already exists

r+ Open a file for read/write.

w+ Open a file for read/write. Erases the contents of the file or creates a new file if it doesn't exist.

a+ Open a file for read/write. The existing data in file is preserved. Creates a new file if the file doesn't exist

PHP Create/Write File

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "aditya pandey \n";
fwrite($myfile, $txt);
$txt = "Hello aditya \n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```