GCA (S5) 02
Exam. Code: WT

Web Technology

SEMESTER V

BACHELOR OF COMPUTER APPLICATION

Block 2



KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

Subject Experts

Prof. Anjana Kakati Mahanta, Gauhati University

Prof. (Retd.) Pranhari Talukdar, Gauhati University

Dr. Jyotiprokash Goswami, Associate Professor, Assam Engineering College

Course Co-ordinators

Dr. Tapashi Kashyap Das, Assistant Professor, KKHSOU

Ms. Sruti Sruba Bharali, Assistant Professor, KKHSOU

SLM Preparation Team

UNITS	CONTRIBUTORS
8, 11 & 12	Ms. Jonalee Barman Kakaty, NERIM
9	Mr. Atowar Islam, Cotton University, Guwahati
10	Ms. Gitimoni Talukdar, Royal Global University, Guwahati
13	Ms. Daisy Kalita, USTM
14 & 15	Mr. Arabinda Saikia, KKHSOU

Editorial Team

Content : Dr. Sanjib Kumar Kalita, Gauhati University

Structure, Format & Graphics: Dr. Tapashi Kashyap Das, KKHSOU

June, 2019 ISBN: 978-93-89123-30-2

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State University is made available under a Creative Commons Attribution-Non Commercial-ShareAlike4.0 License (International): http.//creativecommons.org/licenses/by-nc-sa/4.0.

Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University.

Headquarters : Patgaon, Rani Gate, Guwahati-781 017

City Office : Housefed Complex, Dispur, Guwahati-781 006; Web: www.kkhsou.in

The university acknowledges with thanks the financial support provided by the **Distance Education Bureau, UGC** for the preparation of this study material.

BACHELOR OF COMPUTER APPLICATION

Web Technology

Block-2

CONTENTS

		Pages
UNIT 9:	**Introduction to JavaScript** Introduction, Language Elements: Identifiers, Expressions, Operators, Statement Functions	**149-164**
UNIT 10:	**Objects in JavaScript** Objects: Window, Document, Form Objects, TextBox, TextArea, Button, Radio Button, CheckBox, Select Objects, Date, Match, String Objects; Regular Expression, Arrays	**165-183**
UNIT 11:	**XML** XML – Declaration, Root Element, Child Elements, Element Attributes, Entity References, Comments	**184-195**
UNIT 12:	**AJAX Basics** Introduction, AJAX, XML http Request Object, AJAX Request, AJAX Response, AJAX Events	**196-204**
UNIT 13:	**Introduction to PHP** Server Side Programming , Introduction to PHP, Basic Programming Concepts of PHP: Variables, Data-types, Constants, Scope of Variables, Type of Variables, Type Casting, Operators, Operators Precedence, References	**205-229**
UNIT 14:	**Control statements in PHP** Control Structures: Branching, If statement, Switch statement; Looping: for Loop, while Loop, do while Loop, for each Loop	**230-250**
UNIT 15:	**Functions in PHP** Functions: User Defined Functions, Built-in Function, Functions for Variables; Script Controlling Functions, Array Functions, Date and Time Functions, Mathematical Functions, String Functions, PHP Server Variables; Working with form, Uploading files to Web Server using PHP	**251-276**

BLOCK INTRODUCTION

This is the second block of the course ***Computer Graphics***. This block begins with clipping algorithms. The primary use of clipping in computer graphics is to remove objects, lines, or line segments that are outside the viewing pane. It also focuses on projections and three-dimentional object representation and three-dimentional viewing. Concepts of animation and its applications are discussed at the end of this block.

This block comprises the following eight units:

Unit 8 deals with line and polygon clipping algorithms.

Unit 9 describes three-dimentional object representation. Polygon serfaces, spline serfaces, basic illuminations model etc.are introduced in this unit.

Unit 10 is on Bezier and B-spline curves. Curves are needed for drawing curves many times to draw an object.

Unit 11 is on projections. Parallel and perspective projections etc. are discussed in this unit.

Unit 12 focuses on three-dimentional geometric transformation. General matrix transformation equation in 3D space, translation, scaling and rotation in 3D, composition of 3D transformations are covered in this unit.

Unit 14 is on animation. Animation is a method in which pictures are manipulated to appear as moving images. This unit includes the basic principles of animation, terms associated with animation, types of animation etc.

Unit 15 is on designing of animation. Design of animation sequence, morphing techniques, apllication and future of animation are discussed in this unit.

Each unit of this block includes some along-side boxes to help you know some of the difficult, unseen terms. Some “EXERCISES” have been included to help you apply your own thoughts. You may find some boxes marked with: “LET US KNOW”. These boxes will provide you with some additional interesting and relevant information. Again, you will get “CHECK YOUR PROGRESS” questions. These have been designed for you to self-check your progress of study. It will be helpful for you if you solve the problems put in these boxes immediately after you go through the sections of the units and then match your answers with “ANSWERS TO CHECK YOUR PROGRESS” given at the end of each unit.

UNIT 9: INTRODUCTION TO JAVASCRIPT

UNIT STRUCTURE

9.1 Learning Objectives

9.2 Introduction

9.3 Language Elements: Identifiers

9.4 Expressions

9.5 Operators

9.6 Statement

9.7 Functions

9.8 Let Us Sum Up

9.9 Further Reading

9.10 Answers to Check Your Progress

9.11 Model Question

9.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn how to use JavaScript in a HTML document
- learn about the basic rules for writing a identifier
- understand the use of expressions in JavaScript
- learn the different types of operators used in an expression
- learn and apply the control flow statements and loops in JavaScript programs
- use the concept of function to break up a large program into a number of small and manageable blocks.

9.2 INTRODUCTION

JavaScript is a dynamic computer programming language released in the year 1995 by Netscape and Sun Microsystems. It is a lightweight, interpreted programming language and it creates dynamic web pages that are not static HTML, but contain programs that interact with the user. Uses of JavaScript are:

- Trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.
- Can show, hide, change, resize images, create image rollovers, and create scrolling text across the status bar.
- Interact with the user.
- It can do some processing of forms and can validate user input when the user submits the form.

The following points are required to be remembered:

- JavaScript code is case sensitive
- White space between words and tabs are ignored
- Line breaks are ignored except within a statement
- JavaScript statements end with a semicolon;

Writing a JavaScript Program: JavaScript can be implemented using JavaScript statements that are placed within the <script>... </script> HTML tags in a web page. The <script> tags, containing JavaScript code, can be placed anywhere within the web page, however it is usually recommended to keep it within the <head> tags.

The browser program interprets all the code between <script> tags as a script. The script tag takes two important attributes:

- Language: This attribute specifies the scripting language to be used. In general, its value will be JavaScript.
- Type: This attribute indicates the scripting language in use. The value is set to “text/javascript”.

JavaScript syntax appears as follows:

```
<script language=”javascript” type=”text/javascript”>
JavaScript code
</script>
```

First JavaScript Code

```
<html>
<head>
<script language=”javascript” type=”text/javascript”>
```

```
<!-
document.write ("My First JavaScript Code")
//->
</script>
</head>
<body>
</body>
</html>
```

In the above example ***<!–*** and ***//–>*** is added in case a browser does not support JavaScript. Here ***"//"*** signifies a comment in JavaScript. The function ***document.write( )*** writes a string into HTML document.

9.3 LANGUAGE ELEMENTS: IDENTIFIERS

Let us consider the following statements in a JavaScript program:

```
var student_marks=75
var sum
var flag=0
```

Here, student_marks, sum, flag are unique names that are used as variables also called identifiers.

JavaScript defines certain rules for identifiers which are as follows:

- An identifier must begin with a letter
- An Identifiers can also begin with underscore (_)
- An identifier can also contain letters, digits and underscores.
- An identifier is case sensitive (for example sum and SUM are different identifiers).
- Keywords (Reserved words) in JavaScript cannot be used as identifier.

9.4 EXPRESSIONS

A JavaScript expression is a unit of code that resolves to a value. There are two types of expressions:

- One that assign value to a variable using assignment operator (=)
  For example sum=0 or name='John'
- And one that evaluates expression in the right hand side of assignment operator (=) and assign value to a variable.
  For example: sum=5+6.

Also grouping operator ( ) can be used to control the order of evaluation in an expression. For example in the statement "***total = x * (z + y)***" addition is overriding multiplication.

```
<html>
<head>
<title>Expression Example program</title>
<script type="text/javascript">
<!-
var total, x = 10, y = 20, z = 5;

total = x * z + y;
document.write(total);   // by default 70

total = x * (z + y);
document.write(total);   // by overriding 250

total = x * z + y * z;
document.write(total);   // by default 150
//->
</script>
</head>
</html>
```

9.5 OPERATORS

JavaScript supports the following types of operators.

- **Arithmetic Operator:**
  Addition (+), Multiplication (*), Subtraction (–), Division (/), Modulus (%)
  For example:

```
var a = 30;
var b = 10;
var c;

c = a + b ; // returns 40
c = a - b ; // returns 20
c = a * b ; // returns 300
c = a / b ; // returns 3
c = a % b ; // returns 0
```

- **Comparison Operators:**

  Greater than (>), Less than (<), Greater than or equal to (>=), Less than or equal to (<=), Not equal (!=), Equal (==)

  For example:

```
var a = 1;
var b = 0;
var c = '1';
var d = 2;

a == b ;   // returns false
a != b ;   // returns true
a > d ;    // returns false
a <= c ;   // returns true
```

- **Logical Operators:**

  Logical OR (||), Logical AND (&&), Logical NOT (!)

  For example:

```
var a = true;
var b = false;
var c = true;

a || b ;   // returns true
a && b ;   // returns false
a && c ;   // returns true
```

- **Assignment Operators:**

  = (Simple Assignment), += (Add and Assignment), -= (Subtract and Assignment), *= (Multiply and Assignment), /= (Divide and Assignment), %= (Modules and Assignment)

  For example:

```
var a = 1;
a += 1;    // same as 'a = a + 1' returns 2
a -= 1;    // same as 'a = a  - 1' returns 0
a *= 1;    // same as 'a = a  + 1' returns 1
```

- **Ternary Operator:**

  if Condition is true ? Then value X : Otherwise value Y

  For example:

```
var a = true;
var b = false;
var c;

(a==b)?c=10:c=20

document.write (c); // will print 20
```

- **Increment and Decrement Operator:**

  For example:

```
var x = 5;
var y = ++x; // pre-increment : x equals 6; y equals 6
var p = 5;
var q = p ++; // post increment : q equals 5; p equals 6
```

9.6 STATEMENT

Sometimes we want to run a block of code under certain conditions. Flow control also called conditional statements lets user run code only under certain conditions by means of *if and else* blocks.

Flow Control Statement:

```
<html>
<head>
```

```
<title></title>
<script type="text/javascript">
<!-
var a = true ;
var b = false ;

// Flow control statement

if(b)
document.write('Hello Mr. B'); // this part of the
code will never run
else
document.write('Condition is false');

if(a)
document.write('Hello Mr. A'); // this part of the
code will print
  'Hello Mr. A'
else
document.write('Condition is false');
//->
</script> </head> </html>
```

Switch Statements: A Switch statement can be used in place of nested if statements. It looks at the value of a variable or expression, and run different blocks of code depending on the value.

```
choice='one';
switch ( choice ) {
case 'one':
    alert (' the value selected is ONE ');
    break ;
case 'two':
    alert (' the value selected is TWO ');
    break ;
default :
```

```
        alert (' Select any number between 0 and 1 ');
        break ;
    }
```

JavaScript Loop Statement: Loops permit us to run a block of code a certain number of times.

The for Loop:

Syntax:

for (initialization ; condition ; iteration)

[loop_Body]

- Initialization variable with a value, this statement is executed only once, just before the loop starts.
- Condition statement is executed before each iteration and decides to loop or not by returning a value.
- The iteration statement is executed at the end of each iteration and generally involves decrementing or incrementing a counter.
- The loop_Body contains statements that run on every iteration. To execute multiple statements they are wrapped in a block using curly braces.

Example of a for loop:

```
for ( var k=0; k<10 ; k++) {
// This block will be executed 10 times
document.write(' Iteration number: ' + k +'<br>');
```

The While Loop: A while loop each and every statement inside the loop_Body until the condition evaluates to false.

Syntax:

```
while ([ condition ]) [ loop_Body ]
```

Example of a while loop:

```
    var i = 0;
    while (i < 10) {
    // This block will be executed 10 times
```

```
    document.write(' Iteration number : ' +
    i +'<br>');i++; // increment i
    }
```

The do-while Loop: Same as the while loop, except that the loop's body is executed at least once before the condition is tested.

```
do [ loop_Body ] while ([ conditionl ])
```

Example of a do-while loop:

```
do {
document.write('this loop's body will still executes
only once .');
} while ( false );
```

Break and Continue Statements: A break statement within a loop's body is used to stop the loop.

```
    for ( var k=0; k<10 ; k++) {
    if (k==5) break; //
    document.write(' Iteration number: ' + k +'<br>');
    }
```

Output:

Iteration number: 0

Iteration number: 1

Iteration number: 2

Iteration number: 3

Iteration number: 4

A continue statement within a loop's body is used to skip to the next iteration of a loop.

```
    for ( var k=0; k<10 ; k++) {
    if (k==5) continue; //
    document.write(' Iteration number: ' + k +'<br>');
    }
```

Output:

Iteration number: 0
Iteration number: 1
Iteration number: 2
Iteration number: 3
Iteration number: 4
Iteration number: 6
Iteration number: 7
Iteration number: 8
Iteration number: 9

9.7 FUNCTIONS

A function is a block of reusable code that can be called anywhere in the program. This eliminates the requirement of writing the same code over and over again. A function allows a programmer to break up a large program into a number of small and manageable functions. A function in JavaScript is defined by using the keyword ***function***, followed by a unique function name, list of parameters and a set of statements surrounded by curly braces. A Function can take zero or more arguments, and can optionally return a value.

Syntax:

```
<script type="text/javascript">
<!-
function function_name(list of parameters)
{
Statements
}
//->
</script>
```

Calling a Function: To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
<head>
<script type="text/javascript">
function function_call()
{
document.write ("Program to test a function call");
}
</script>
</head>
<body>
<p>Click the following button to  test a function
call</p>
<form>
<input type="button" onclick="function_call()"
value="CLICK HERE">
</form>
</body>
</html>
```

Output:

Click the following button to test a function call



Function with Arguments:

```
<html>
<head>
<script type="text/javascript">
function function_call(string1,string2,year)  //
function with arguments
{
document.write ("Program to test a function with
arguments<br>");
document.write (string1+string2+' '+year);
}
```

```
</script>
</head>
<body>
<p>Click the following button to test a function
with arguments</p>
<form>
<input type="button" onclick="function_call
('JAVA','SCRIPT', 2018)" value="CLICK HERE">
</form>
</body>
</html>
```

Output:

Program to test a function with arguments

JAVASCRIPT 2018

Return Statement: Return statements are used to return values from a function.

```
<html>
<head>
<script type="text/javascript">
function sum(a, b)
{
var result;
result = a + b;
return result; // return statement
}
function show()
{
var result;
result = sum(9, 8);
document.write (result );
}
show(); //calling the function show() ,output is 17
</script>
</head>
```



CHECK YOUR PROGRESS

Q.1: Which of the following is nor a valid identifier

a) sum_9 b) _34p

c) 9abc d) Rty_9sd

Q.2: Inside which HTML tag JavaScript code is written?

a) <js> b) <script>

c) <scripting> d) <javascript>

Q.3: A JavaScript code is written inside:

a) The <title> section b) The <head> section

c) The <body> section

d) Both the <head> section and the <body> section are correct

Q.4: Which of the following is a valid *if statement* ?

a) if condition then statement

b) if condition elseif statement

c) if condition statement

d) if condition else statement

Q.5: How do you create a function '*prime'* in JavaScript?

a) prime( ) b) function:prime()

c) function = prime() d) function prime()

Q.6: Which of the following is valid FOR loop?

a) for (i = 1 to 10) b) for (i =10; i >= 10; i—)

c) for (i <= 10; i++) d) for (i = 0; i <= 10; i+1)

Q.7: To insert multiline comment in JavasScript:

a) //This is a Multiline comment// b) <!—This is a multiline comment–>

c) /*This is a Multiline comment*/ d) All of the above

Q.8: JavaScript is also known as .....

a) Client Side Scripting Language

b) Server Side Scripting Language

c) Scripting Language d) All of the above

Q.9: The output of the following JavaScript code is .....

```
<script type="text/javascript">
x=8+"8";
document.write(x);
</script>
```

a) 16 b) 88 c) 8 d) Error

Q.10: A ..... statement within a loop's body is used to ..... to the next iteration of a loop.

a) Break, Skip b) Continue, Break

c) Continue, Skip d) None of the above



9.8 LET US SUM UP

- JavaScript released in the year 1995 by Netscape and Sun Microsystems is a lightweight, interpreted programming language used to create dynamic web pages.
- It can do some processing of forms and can validate user input when the user submits the form.
- A JavaScript identifier must begin with a letter/underscore followed by letters/digits/underscores.
- A JavaScript identifier is case sensitive and keywords cannot be used as identifier
- Grouping operator ( ) can be used to control the order of evaluation in an expression
- JavaScript supports a variety of operators like arithmetic, logical, comparison, assignment and ternary.
- JavaScript supports conditional statements like if-else and switch.
- JavaScript supports Loop control statements like for-loop, do-while loop and while loop.
- A function can be invoked in a javascript by simply writing the name of the function with the list of arguments.
- Return statements are used to return values from a function.



9.9 FURTHER READING

1) Crockford, Douglas. (2008). *JavaScript: The Good Parts*. Shroff Publishers & Distributors.
2) *JavaScript: The Definitive Guide*. (Edition 2) (1996). Publisher: O'Reilly Media.
3) Haverbeke, Marijn. *Eloquent JavaScript: A Modern Introduction to Programming*. (2nd Edition).
4) Sharma, Kumar Chetan; Stefanov, Stoyan; Antani, Ved; Hillar, Gastón C. *JavaScrip: Object-Oriented Programming*. Publisher: Packt Publishing Limited.



9.10 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q. No. 1: (c) **Ans to Q. No. 2:** (b)
Ans to Q. No. 3: (d) **Ans to Q. No. 4:** (c)
Ans to Q. No. 5: (d) **Ans to Q. No. 6:** (b)
Ans to Q. No. 7: (c) **Ans to Q. No. 8:** (a)
Ans to Q. No. 9: (b) **Ans to Q. No. 10:** (c)



9.11 MODEL QUESTIONS

Q.1: What is JavaScript? Discuss a few features of JavaScript. What are the advantages of using JavaScript?

Q.2: Explain the rules for naming a variable in JavaScript?

Q.3: How to define and call a function in JavaScript? How does the return statement in a function call work?

Q.4: What is the use of 'this' operator in JavaScript? Write a JavaScript to explain its working.

Q.5: What is the scope of a variable in JavaScript?

Q.6: Write a JavaScript program to convert temperature from Celsius to Fahrenheit.

Q.7: Write a JavaScript program which prompts user to guess a number between one to twenty. The program then generates a random integer between one and twenty. If both the number matches, the program will display a message “YOU WIN” otherwise display a message “YOU LOST”

Q.8: Write a JavaScript program to check the average marks of a student in five subjects. The program will print the grade based on the following table:

GRADE	MARKS RANGE
A+	91-100
A	81-90
B+	71-80
B	61-70
C	51-60
F	0-50

Q.9: Write a JavaScript program which iterates the integers from 1 to 50. But for multiples of four and five print “X” and “Y” respectively. For numbers which are multiples of both four and five print “XY”.

Q.10: Write a JavaScript program to find the Armstrong numbers. For example, 153 is an Armstrong number since 1**3 + 5**3 + 3**3 = 153.

Q.11: Write a JavaScript function that reverses a number. For example input:1578 , output:8751

Q.12: Write a JavaScript function that accepts a number as a parameter and check the number is prime or not.

*** ***** ***

UNIT 10: OBJECTS IN JAVASCRIPT

UNIT STRUCTURE

10.1 Learning Objectives

10.2 Introduction

10.3 Objects: Window, Document

10.4 Form Objects

- 10.4.1 TextBox
- 10.4.2 TextArea
- 10.4.3 Button
- 10.4.4 Radio Button
- 10.4.5 CheckBox
- 10.4.6 Select Objects

10.5 Date, Math, String Objects

10.6 Regular Expression

10.7 Arrays

10.8 Let Us Sum Up

10.9 Answers to Check Your Progress

10.10 Further Reading

10.11 Model Questions

10.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn the concept of objects in JavaScript
- create a new window and dialog boxes in JavaScript
- make smarter HTML pages with JavaScript Form objects
- apply Date, Math and String Objects in different applications of JavaScript.
- describe search patterns with the help of regular expressions in JavaScript
- apply the concept of arrays in JavaScript.

10.2 INTRODUCTION

JavaScript also supports the concept of objects. Objects are a way of organizing the variables. For example screen elements such as text boxes, buttons, forms, images, and are treated as objects. Every object has one or more properties and methods.

- Property defines the characteristics of an object. For example, name, length, weight, height, color etc.
- Methods are the actions that the object can perform. For example write, open, close, alert, confirm.

An object is referred to by using ***objectName***. Its property is referred to by using ***objectName.propertyName*** and any method of an object can be accessed by ***objectName.methodName()***.

10.3 OBJECTS: WINDOW, DOCUMENT

Window Object: The window object can be used to create new windows and dialog boxes and it includes the following methods:

- Open – Opens a new window
- Close – Closes the window
- alert – Displays an alert message box
- prompt – Displays a prompt dialog box
- confirm – Displays a confirm dialog box

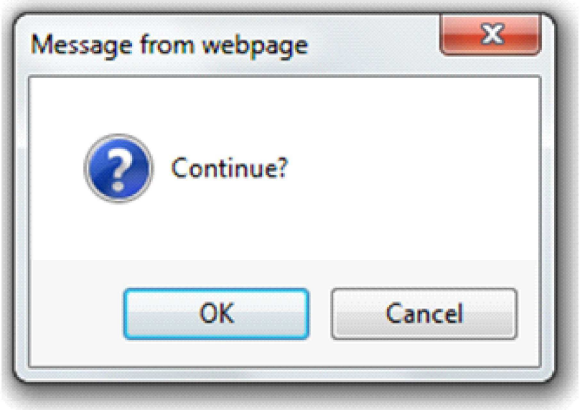
Alert Message Box: The alert message box displays a simple message.

alert('Hello world');



Confirm Dialog Box: The confirm dialog box displays a confirmation type message and then returns either a true or false value depending on the button being pressed.

```
var choice = confirm("Continue?");
document.write(choice);
```



Prompt Dialog Box: The prompt dialog box provides a way of getting input from the user. The prompt function has two arguments. The first is the prompt message and the second is a default value, if any.

```
var choice = prompt("Name:","");
document.write(choice);
```



Document Object: When a web browser loads an HTML document, it becomes a document object. Some of the properties and methods of document object are:

Property	Description
getElementById()	used to get the element that has the ID attribute with the specified value
title	used to get or set the title of the document
URL	used to get the URL of the HTML document
write()	used to writes HTML expressions/code to a document
writeln()	used to as write HTML expressions/code to a document with newline

10.4 FORM OBJECTS

JavaScript makes HTML Forms “smarter” by utilizing its decision-making capabilities. Using JavaScript one can process simple forms with or without invoking the server. In a JavaScript enhanced forms, objects take care of all the basic requirements such as taking or setting inputs. One main difference between a HTML form and a JavaScript-enhanced form is that a JavaScript form depends on event handlers (e.g., onClick or onSubmit) to invoke a user defined action in the form such as clicking/submitting a button. Typical form control objects are:

- Text box Object
- Text Area Object
- Button Object
- Radio buttons Object
- Check boxes Object
- Select Object

10.4.1 Text Box Object

A text box is a form control element used to input text information.

Text Box Object Properties:

Property	Description
Value	Used to get or set the value of the text box.
defaultValue	Used to get or set the default value of the text box.
maxLength	Used to get or set the maximum number of characters in a text box.
Name	Used to get or set the value of the name attribute of a text box.
readOnly	Used to get or set the text box to read-only.
Type	Used to get returns type of form element of a text box.
Size	Used to get or set the width of a text box.

In the example below the event handler ***onClick = “textbox_get_set()*** is used to get input from textbox1 using ***test_value = form1.textbox1.value*** and set it to textbox2 using

form1.textbox2.value = test_value. Here *form1*, *textbox1*, *value* is the name of the form, text box and property of the object respectively.

```
<html>
<head>
<title>JavaScript Text Box Object Example</title>
<script type=‘‘text/javascript”>
<!–
function textbox_get_set( ) {
   var test_value = form1.textbox1.value; //get value
   from textbox 1
   form1.textbox2.value=test_value;    // set value
   to textbox 2
}
//->
</script>
</head>
<body>
<form name="form1" action=""method="GET">
Enter a value in the TextBox-1:
<input type="text" name="textbox1" value=""><P><BR>
<input type="button" name="button" Value="Click"
onClick="textbox_get_set()"><BR><BR>
Value copied from TextBox-1: <input type="text"
name="textbox2" value=""><P>
</form></body></html>
```

10.4.2 Text Area Object

Text areas are used for multiple-line text entry and are created using COLS and ROWS attributes.

Text Area Object Properties:

Property	Description
Value	Used to get or set the value of the text area.
defaultValue	Used to get or set default value of a text area.

Type	Used to get or set the returns type of form element of a text area.
Name	Used to get or set the value of the name attribute of a text area.
Rows	Used to display the number of rows in a text area.
Cols	Used to display the number of columns in a text area.

In the example given below a text area with 60 characters wide by 7 rows is created. On clicking the button ***click*** the event handler reads the input from the Text Area and displays it using the alert window.

```
<html>
<head>
<title>JavaScript Text Area Object Example</title>
<script type="text/javascript">
<!—
function mytextarea (form) {
    alert (form.textarea1.value);
}
//->
</script>
</head>
<body>
<form name="form2">
Type Something:<textarea name="textarea1" COLS="60"
ROWS="7"></TEXTAREA>
<br><input type="button" name="button_click"
Value="Click" onClick="mytextarea(this.form)">
</form>
</body>
</html>
```

10.4.3 Button Object

Button object represents a button (<input type="button">) within a HTML form. It can be accessed and manipulated using JavaScript Button object.

Events	Description
onClick	When the user clicks a button a Code is executed.

10.4.4 Radio Button Object

Radio object represents a radio button in an HTML form.

Radio Object Properties:

Property	Description
Checked	Used to get or set the checked state of a radio button.
defaultChecked	Used to get the default value of the checked attribute.
Name	Used to get or set the value of the name attribute of a radio button.
Value	Used to get or set the value of the radio button.
Type	Used to get the type of element which is radio button.

Radio Object Methods:

Method	Description
click()	Invokes if the user clicked the button.
blur()	Shifts focus away from the radio button.
focus()	Moves focus to the radio button.

In the example given below the statement ***document.getElementById("myRadio").checked*** returns the value *true* if the radio button is checked otherwise false. If it is true the radio button value is displayed using the statement **alert*(document.getElementById("myRadio").value);***

```
<html><head><title>Radio Button Object Example</title>
<script>
function myRadioFunction(form) {
var x = document.getElementById("myRadio").checked;
```

```
if(x)
alert(document.getElementById("myRadio").value);
}
</script></head><body>
<form name="radio_form">
<input type="radio" name="gender" value="male"
id="myRadio"> MALE<br>
<input type="button" name="button_click"
Value="CLICK"
onClick="myRadioFunction(this.form)"></button>
</form></body></html>
```

10.4.5 Checkbox Object

Check box object represents a checkbox in an HTML form. It allows the user to select one or more options from the available choices.

Checkbox Object Properties:

Property	Description
checked	Used to get or set the checked state of a checkbox.
Value	Used to get or set the value of the checkbox attribute.
Name	Used to get or set the name of the checkbox.
Type	Used to get the type of “check”.

Checkbox Object Methods:

Method	Description
click()	Used to set the checked property.

In the example given below the statement ***document.getElementById(“myCheckbox”).checked*** is used to check if the checkbox is checked? If true the value of the checkbox is returned using the statement ***document.getElementById(“myCheckbox”).value*** and is displayed in the HTML document.

```
<html><head><title>Check Box Object Example</title>
<script>
```

```
function myCheckboxFunction(form) {
var x = document.getElementById("myCheckbox").checked;
var y = document.getElementById("myCheckbox").value;
if(x)
document.getElementById("cbox").innerHTML=y;
else
document.getElementById("cbox").innerHTML="Checknox
unchecked";
}
</script>
</head><body>
<form name="checkbox_form">
<input type="Checkbox" value="APPLE"
id="myCheckbox"> APPLE <br>
<input type="button" name="button_click"
Value="CLICK"
onClick="myCheckboxFunction(this.form)"></button>
<p id="cbox"></p>
</form></body></html>
```

10.4.6 Select Object

Select object represents a dropdown list in an HTML form. It allows the user to select one or more options from the available choices.

Select Object Collections:

Collection	Description
Options	Used to get list of all the options in a dropdown list.

Select Object Properties:

Property	Description
Length	Used to get the number of options in a dropdown list.
selectedIndex	Used to get or set the index of the selected option in a dropdown list.
Type	used to get the type of form element.
name	used to get the name of the selection list.

Select Object Methods:

Method	Description
add()	Used to add an option to a dropdown list.
remove()	Used to remove an option from a dropdown list.

In the example given below the statement ***document.getElementById("mySelectList").length*** is used to get the number of options in a dropdown list ***mySelectList***.

```
<html><head><title>Select Object Example</
title><script>
function mySelectListFunction(form) {
var y = document.getElementById("mySelectList").length;
document.getElementById("selectList").innerHTML=y;
}
</script></head><body>
<form name="selectList_form">
<select id="mySelectList" size="3">
<option>HONDA</option>
<option>MARUTI</option>
<option>TOYOTA</option>
</select>
<input type="button" name="button_click" Value=
"CLICK" onClick="mySelectListFunction(this.form)"
></button>
<br><br>No. of elements in the list is:<p
id="selectList"></p>
</form></body></html>
```

10.5 DATE, MATH, STRING OBJECTS

The JavaScript Math object provides several useful methods:

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x (radians)
asin(x)	Returns the arcsine of x, in (radians)

atan(x)	Returns the arctangent of x as a value
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (radians)
exp(x)	Returns the value of Exponent of x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

Date Object: Date methods can be used to get and set values like years, months, days, hours, minutes, seconds, milliseconds:

Method	**Description**
getDate()	Get the day as a number(1-31)
getDay()	Get the weekday as a number(0-6)
getFullYear()	Get the four digit year
getHours()	Get the hour(0-23)
getMinutes()	Get the minutes (0-59)
getSeconds()	Get the seconds(0-59)
getMonth()	Get the month(0-11)
getTime()	Get the time (milliseconds since January 1m 1970)

Example:

```
<html>
<head>
<script type="text/javascript">
 var d=new Date();
document.write('Today\'s Date is : ' + d.getDate()
+ '-' + d.getMonth()+ '-' + d.getFullYear() );
</script>
</head>
```

String Object: The String object provides methods for string manipulation and formatting:

Method	Description
search()	Returns the position of the match
indexOf()	Returns index of the first occurrence of a specified text in a string
lastIndexOf()	Returns index of the last occurrence of a specified text in a string
substring(start,end)	Extracts a part of a string
substr(start,length)	Extracts a part of a string
replace()	Replaces a specified value with another value in a string
toUpperCase()	String converted to uppercase
toLowerCase()	String converted to lowercase
concat()	To concatenate two strings
charAt()	Returns the character at specified index in a string

Example:

```
<script>
var str='the brown fox quickly jumps over the lazy dog'
document.write('First Index of \'the\' is:' +
str.indexOf('the') +'<br>');  // 0utput is 0
document.write('Last Index of \'the\' is:' +
str.lastIndexOf('the') +'<br>');  //output is 33
document.write('SubString  from index 22 to 25 is:'
+ str.substr(22,4) +'<br>');    // jump
</script>
```

10.6 REGULAR EXPRESSION

A regular expression is a sequence of characters that forms a **search pattern**. When you search for data in a text, you can use this search pattern to describe what you are searching for. A regular expression can be a single character, or a more complicated pattern.

Expression	Description
[abc]	Matches any of the characters between the brackets
[^abc]	Matches any of the characters not between the brackets
[0-9]	Matches any of the digits between the brackets
(x|y)	Matches any of the alternatives separated with |
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning or at the end of a word
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n
i	Perform case-insensitive matching
g	find all matches rather than stopping after the first match
m	Perform multiline matching

In example given below the pattern ***/[Era]/g*** select all the characters that matches either 'E" or 'r' or 'a' from the string "JavaScript Regular Expression Example".

```
<html><head><title>Javascript RegularExpression
Example</title>
<script>
function Reg_exp_Function() {
var str = "JavaScript Regular Expression Example";
var pattern = /[Era]/g;// select all the characters
that matches either 'E' or 'r' or 'a'
var x = str.match(pattern);
document.getElementById("display").innerHTML = x;
}
</script></head><body>
<input type="button" name="button_click" Value=
"CLICK" onClick="Reg_exp_Function()"></button>
<p id="display"></p>
</body></html>
```

10.7 ARRAYS

Arrays are fixed size collection of related items of the same type. In its simplest form, it can be used to represent list of numbers or a list of strings etc.

A Simple Array:

```
var numbers = [ 'one', 'two', 'three', 'four' ];
var num = [ 16, 2 , 77, 90, 12 ];
```

Accessing Array items by Index:

```
var numbers = [ 'one', 'two', 'three', 'four' ];
document.write(numbers[2]);// displays 'three'
```

Testing the Size of an Array:

```
var numbers = [ 'one', 'two', 'three', 'four' ];
document.write(numbers.length)    // output is 4
```

Changing the Value of an Array Item:

```
var numbers = [ 'one', 'two', 'three', 'four' ];
numbers[2] = '3'; // 'three' is replaced by '3'
in the array
```

Adding Elements to an Array:

```
var numbers = [ 'one', 'two', 'three', 'four' ];
numbers.push('five'); // is used to add elements
for ( var k=0; k<5 ; k++) {
document .write(numbers[k]+'');
}
```

Output:

one two three four five

Working with Arrays:

```
var numbers = 'JAVASCRIPT';
split_num=numbers.split(''); // splits the string
JAVASCRIPT into individual character
for ( var k=0; k<5 ; k++) {
```

```
document.write(split_num[k]+'<br>');
}
document.write(split_num.join('')); // joins the
spitted string into a single string
```

Output:

J

A

V

A

S

JAVASCRIPT



CHECK YOUR PROGRESS

Q.1: The correct syntax to change the content of the HTML element given below in a JavaScript is:

JAVASCRIPT

a) document.innerHTML = "Hello World!";

b) document.getElementById(p id="test").innerHTML = "HTML";

c) document.getElementById("test").innerHTML = "HTML";

d) document.getElementByName("test").innerHTML = "HTML";

Q.2: To write "INDIA" in an alert box?

a) msgBox("INDIA"); b) msg("INDIA");

c) alert("INDIA"); d) alertBox("INDIA");

Q.3: What is the correct way to write a JavaScript array?

a) var colors = ("pink", "white", "black")

b) var colors = ["pink", "white", "black"]

c) var colors = (1:"pink", 2:"white", 3:"black")

d) var colors = "pink", "white", "black"

Q.4: How to find the largest number between A and B?

a) Math.Larg(A,B) b) Math.ceil(A,B)

c) Math.Highest(A,B) d) Math.max(A,B)

Q.5: When a user clicks on an HTML element, the vent that occurs is?

a) onclick b) onchange

c) onmouseclick d) Onmouseover

Q.6: The built-in string method that returns character at the specified index is?

a) getCharAt() b) characterAt()

c) charAt() d) None of the above.

Q.7: What does /[^0-9]* regular expression indicate ?

a) match zero or more characters that are digits

b) match zero or more characters that are not digits

c) match one or more characters that are not digits

d) match one or more characters that are digits

Q.8: The URL property belongs to _____ object.

a) Window b) String

c) Form d) Document

Q.9: How a user can be asked for input as well as shown a message?

a) alert b) confirm

c) prompt d) inputMsg

Q.10: "this" keyword in javascript refers to .....?

a) previous object b) current object

c) variable d) None of the above



10.8 LET US SUM UP

- JavaScript supports the concept of objects which are a way of organizing the variables.
- Screen elements in HTML web page such as text boxes, buttons, forms, images, and are treated as objects.

- Every object has one or more properties which define the characteristics of an object and methods which define the actions that the object can perform.
- JavaScript window object is used to create new windows and dialog boxes using the methods: Open, Close, alert, prompt and confirm.
- JavaScript form control objects are: Text box, Text Area, Button, Radio buttons, Check box, Select.
- JavaScript supports Date, Math and String Objects.
- A JavaScript regular expression is a sequence of characters that forms a search pattern used to search for data in a text.
- JavaScript also supports the concept of Arrays which are fixed size collection of related items of the same type such as list of numbers or a list of strings.



10.9 FURTHER READING

1) Crockford, Douglas. (2008). *JavaScript: The Good Parts*. Shroff Publishers & Distributors.
2) *JavaScript: The Definitive Guide*. (Edition 2) (1996). Publisher: O'Reilly Media.
3) Haverbeke, Marijn. *Eloquent JavaScript: A Modern Introduction to Programming*. (2nd Edition).
4) Sharma, Kumar Chetan; Stefanov, Stoyan; Antani, Ved; Hillar, Gastón C. *JavaScrip: Object-Oriented Programming*. Publisher: Packt Publishing Limited.
5) Stefanov, Stoyan. *Object-Oriented JavaScript*.



10.10 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q. No. 1: (c) **Ans to Q. No. 2:** (c)

Ans to Q. No. 3: (b) **Ans to Q. No. 4:** (d)

Ans to Q. No. 5: (a) **Ans to Q. No. 6:** (c)

Ans to Q. No. 7: (b) **Ans to Q. No. 8:** (d)

Ans to Q. No. 9: (c) **Ans to Q. No. 10:** (b)



10.11 MODEL QUESTION

Q.1: What are Objects in JavaScript? What is Document, Window & Form Object?

Q.2: How is an array represented in JavaScript? Define and initialize two arrays one with string and one with number.

Q.3: Write a built-in method that returns the character at the specified index.

Q.4: Write a built-in method that combines the text of two strings and returns a new string and also returns the length of the string.

Q.5: Write a built-in method that removes the last element from an array and returns that element.

Q.6: Write a built-in method that adds one or more elements to the end of an array and returns the new length of the array.

Q.7: Write a built-in method that returns the calling string value converted to lower case and to upper case.

Q.8: Write a JavaScript program to display the current day and time in the following format.

Today is: Monday.

Current time is: 23: 07: 52

Q.9: Write a JavaScript program to replace a character at the specified position of a given string and return the new string.

Q.10: Design a HTML Form that contains all possible form objects. Write a JavaScript program to show usage of each object.

Q.11: Write a JavaScript program to sort the items of an array in ascending order.

Q.12: Write a JavaScript program to find an element in an array. Also count the number of times the element is present in the array.

array: var arr1=[7, ‘r’, 6, ‘a’, ‘r’, 12, 63, ‘a’, 3, ‘a’, 7, 14, 29, 7];

Output: a (5 times)

Q.13: Write a JavaScript program which takes as input ‘JAVASCRIPT PROGRAM’ and the output is displayed as ‘JavaScript program’.

Q.14: Design a HTML page as shown below. Also write a JavaScript program to validate the fields for example name field cannot include digits, zip code cannot include characters, no field can be left blank and email address should be a valid one on clicking the submit button.



Q.15: Write a JavaScript program to count number of words in string. Use JavaScript regular expression.

Q.16: Write a JavaScript function to check whether a given input is a valid vehicle registration number (e.g., AS-01-2233). Use JavaScript regular expression.

Q.17: Write a JavaScript function to check whether a given input is a valid uniform resource locator (URL). Use JavaScript regular expression.

*** ***** ***

UNIT 11: XML

UNIT STRUCTURE

11.1 Learning Objectives

11.2 Introduction

11.3 Introduction to XML

11.3.1 Uses of XML

11.4 XML Syntax

11.4.1 General Rules for a Well-Formed XML Document

11.5 XML Declaration

11.6 XML Comments

11.7 Tags and Elements

11.7.1 Root and Child Elements

11.8 Element Attributes

11.9 Entity References

11.10 Document Type Definition (DTD)

11.11 Let Us Sum Up

11.12 Further Reading

11.13 Answers to Check Your Progress

11.14 Model Questions

11.1 LEARNING OUTCOMES

After going through this unit, you will be able to:

- learn the basics of XML language
- learn the difference between HTML and XML
- learn the utility of XML
- describe various rules to create a valid and well formed XML document
- learn how to use XML documents in web services.

11.2 INTRODUCTION

The limitation of HTML is that the set of elements and attributes are finite, pre-defined and fixed. They cannot be extended. That means, a

designer of a page cannot incorporate new elements or attributes to the set of pre-existing sets. Moreover, the HTML tags are defined keeping only the presentation in mind. Meaning of data is not associated with a tag. For example, a list simply signifies that it contains a list of items. But whether it is a list of students or a list of fruits cannot be derived from the tags. So, there exists another text based markup language called the eXtensible Markup language (XML), which is also a derivative of SGML. In this unit, we will learn about XML.

11.3 INTRODUCTION TO XML

XML tags identify the data and are used to organize and store data, rather than specifying how to display the data. In XML a user can encode the data with meaningful user-defined tags and attributes. It is designed to describe data and the focus is on what data is, whereas, HTML is about displaying data and the focus is on how data looks like when displayed.

XML has no predefined tags. It simply specifies certain rules on syntax and composition of elements. A user can define its own meaningful tags. XML uses a *Document Type Definition* (DTD) or an XML Schema to describe the grammar of the use of tags for a document or for a type of documents.

An XML document must always be well-formed and valid. With XML we can encode data during storage and transmission with meaningful tags and attributes. But for presentation by a browser we need to transform them to HTML. Therefore, we can use XML as a cross-platform, software and hardware independent tool for encoding data for storage and transmission and use HTML for presentation.

11.2.1 Uses of XML

XML is mainly designed to store, carry, and exchange data. It is not designed to display data. Some of the uses of XML are:

- *XML is used to exchange data:* With XML, data can be exchanged between incompatible systems. In the real world, computer systems and databases contain data in incompatible formats.

One of the most time-consuming challenges for developers has been to exchange data between such systems over the Internet. Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications.

- *XML can be used to share data:* With XML, plain text files can be used to share data. Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data. This makes it much easier to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers.
- *XML can be used to store data:* With XML, plain text files can be used to store data. XML can also be used to store data in files or in databases. Applications can be written to store and retrieve information from the store, and generic applications can be used to display the data.



CHECK YOUR PROGRESS

Q.1: What does XML stand for?

.....

Q.2: What is the basic difference between an HTML and XML document?

.....

.....

.....

Q.3: Does XML have predefined tags?

.....

Q.4: What is the basic use of XML?

.....

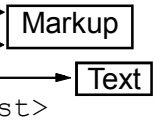
.....

11.3 XML SYNTAX

An XML document is basically organized in a tree based structure with one root node and multiple child nodes. It mainly consists of a markup and the text. An example of a complete XML document is as follows:

Example 12.1:

```
<?xml version= "1.0" encoding = "ISO-8859-1"?>
      <organization>
            <employee>
                  <name>
                        <first> ABC </first>
                        <middle> PQR </middle>
                  <last> XYZ </last>
            </name>
      </employee>
</organization>
```



11.3.1 General Rules for a Well-Formed XML Document

Writing syntax of XML is simple but there are certain definite rules required to be followed. The rules are very easy to learn, and also easy to use. They are as follows:

- The document must start with an XML declaration,

  ```
  <? xml version="1.0"?>
  ```

- There must be only one root element in which all others are contained.
- Each element or tag must end with an end tag

  ```
  <elem>...</elem>
  ```

- All elements must be properly nested. No overlapping is permitted.
- XML tags are case sensitive.
- XML attribute values must be quoted.
- XML comments are written as <!– This is a comment –>.

An XML document is said to be *well-formed* if it conforms to all the XML syntax rules.

11.4 XML DECLARATION

The first line in the document, i.e., the XML declaration (processing instruction), defines the XML version and the character encoding used in the document. An XML declaration helps the parser to parse XML documents. In the above example, the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set.

An XML document must begin with a ***<? xml >*** and it is case-sensitive. So it needs to be written in lower case letters. And this declaration has to be strictly included in the beginning of the document. Also the declaration must include the version of the xml standard used. However, the XML declaration does not needs to be closed as it is not a tag. Some of the examples of an XML declaration are as follows:

- a declaration with only the version

```
<? xml version=”1.0"?>
```

- a declaration with both version and character encoding

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

However there exist an additional parameter called *standalone* with values “yes” or “no” in the xml declaration which informs the parser whether the document depends on some external source for its information. The default value of it is “no”.

11.5 XML COMMENTS

Comments are mainly added as notes so that the code can be understood properly. They enhance the readability of the code. Any kind of related information can be included as a comment. XML comments are similar to HTML comments and remain invisible in the XML code. A comment starts with **<!–** and ends with **–>**. The syntax to include an XML comment is as follows:

```
<!—Comment 1—>
<!—Comment 2—>
```

Comments may appear anywhere in a document but not before the declaration. An example of an XML comment is as follows:

```
<?xml version= "1.0" encoding = "ISO-8859-1"?>
<!— An XML document on the structure of an
organization—!>
```



CHECK YOUR PROGRESS

Q.5: When can an XML document said to be well-formed?

.....

.....

Q.6: What does XML declaration signify?

.....

.....

Q.7: What does the XML declaration defines?

.....

.....

Q.8: What does "standalone" signify in the XML declaration?

.....

.....

.....

Q.9: What is the syntax to include a comment in a XML document and where can it be included?

.....

.....

.....

11.6 TAGS AND ELEMENTS

Any element written within < > is said to be a **tag**. An XML document can be structured with several user defined tags. The whole structure of an

XML document depends on the use of meaningful tags. In the above example, <organisation>, <name>, <first> are all user defined tags.

For naming a tag, any alpha-numeric character can be used. All these tags are case sensitive. For example, <name> and <nAmE> are two different tags. All tags must have closing tag. If a tag is empty (i.e. without a closing tag), then it should be written as <*tag/*>.

The example above also shows the nesting of tags, i.e. one tag lies within the other. The tag <name> lies within the <employee> tag which signifies the name of an employee.

11.6.1 Root and Child Elements

As already stated, an XML document is organized in a tree-based structure. Like trees, every XML document has one root element or tag (<organization>) and one or many child elements. In the above example, <*organization*> is the root element which has <*employee*> as its child element. The element <*name*> has got three child elements.

11.7 XML ATTRIBUTES

Just like HTML, XML can also have attributes. Attributes are generally used to describe the elements in detail. That is, they provide some additional information about the element. Attributes are always written within a tag. They always hold a single value and the value needs to be written in a double (“ “) or single (‘ ‘) quotation marks. In XML, there is always flexibility that relevant tags or their attributes can be incorporated as per the requirement.

As in HTML, *src, height, width, alt, etc* are some of the attributes of the <*img*> tag to describe it in a better manner. For example, in the above example, <*employee*> tag can have attributes like *id, dept, marital_state, etc* which can be written as follows:

```
<employee id = “101” dept= “Marketing”
marital_state = “single”>
```

11.8 ENTITY REFERENCES

Entities are used to define shortcuts to special characters within the XML documents. They are variables used to define common text. Entity references are references to entities. For example, the HTML entity reference: " " ("no-breaking-space" entity), is used to insert an extra space in a document. Entities are expanded when a document is parsed by an XML parser. Entities are declared as follows:

```
<!ENTITY entity_name "entity_value">
```

The *entity_name* is the name of the entity and *entity*_value is the value assigned to it. Any value here is de-referenced by adding **'&**' before the entity name and **';'** at the end of it. For example, if an entity called "Guwahati" needs to be referred as "ghy" in short throughout the XML document, then it can be done as follows:

```
<!ENTITY Ghy "Guwahati">
```

and can be used later in the document as **&ghy;** wherever needed. This is said to be an **entity reference.** Some of the predefined entities in XML are: < (<), > (>), & (&), " ("), ' ('). The data or text that any element will hold is symbolised by ***PCDATA***. PCDATA means parsed character data which is the text found between the start tag and the end tag of an XML element. It is text that will be parsed by a parser.

Another reference called the **Character References** contain references, such as **A** which has a hash mark ("**#**") followed by a number. The number always refers to the Unicode code of a character. In this case, 65 refer to alphabet "A".

11.9 DOCUMENT TYPE DEFINITION (DTD)

Since tags are not predefined in XML, so there must be a way to validate the tags used. The XML Document Type Definition, commonly known as DTD, is a way to describe XML language precisely. DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language. An XML DTD can be either

specified inside the document, or it can be kept in a separate document and then liked separately. Its basic syntax is as follows:

```
<!DOCTYPE element
[
      declaration 1
      declaration 2
      ……..
]>
```

In the above syntax:

- The **DTD** starts with <!DOCTYPE delimiter.
- An **element** tells the parser to parse the document from the specified root element.
- The square brackets [ ] enclose an optional list of entity declarations called **Internal Subset**.

The main constituents of XML DTD are: elements, attributes, entities, PCDATA, CDATA. *Elements* or tags and *attributes* in the DTD will be those that will make up the XML document. An example of XML DTD is as follows:

Example 12.2:

```
<!DOCTYPE organization [
      <!ELEMENT employee (Name)>
            <!ELEMENT Name (first, middle, last)>
            <!ELEMENT first (#PCDATA)>
            <!ELEMENT middle (#PCDATA)>
      <!ELEMENT last (#PCDATA)>
           <!ATTLIST employee  marital_state
           (married | single) #REQUIRED>
      <!ENTITY kr "KUMAR">
]>
```

A **well-formed** XML document is said to **valid** if it also conforms to the rules specified in the DTD.



CHECK YOUR PROGRESS

Q.10: What is the biggest advantage of an XML tag?

.....

.....

Q.11: Are XML tags case sensitive?

.....

Q.12: Define an attribute? How many values can an attribute hold?

.....

.....

Q.13: Name some pre-defined entities in XML.

.....

.....

Q.14: When can a XML document said to be valid?

Q.15: What is PCDATA?

.....

.....



11.10 LET US SUM UP

- XML is a way of marking up data that separates structure from formatting and style.
- XML can greatly be used to store information about the documents.
- XML have predefined tags attributes.
- All the rules for XML declaration, naming tags, attributes, etc must be strictly followed.
- To validate XML tags, a document type definition (DTD) must be used.
- An XML document must be well formed and valid.



11.11 FURTHER READING

1) Roy, Uttam K. (2010). *Web Technologies*. Oxford Higher Education.
2) Sharma, Pankaj. (2011). *Introduction to Web Technology*. Katson Books.



11.12 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: XML stands for extensible markup language.

Ans. to Q. No. 2: The basic difference between HTML and XML is that HTML is mainly used for presentation of data but XML is used for data storage.

Ans. to Q. No. 3: No, XML does not have predefined tags.

Ans. to Q. No. 4: The basic use of XML is that it is used for storing a large set of data.

Ans. to Q. No. 5: An XML document is said to be well-formed if it conforms to all the XML syntax rules.

Ans. to Q. No. 6: An XML declaration is used as an indicator that helps the parser to parse XML documents.

Ans. to Q. No. 7: An XML document defines the XML version and the character encoding used in the document.

Ans. to Q. No. 8: The “standalone” in XML declaration informs the parser whether the document depends on some external source for its information

Ans. to Q. No. 9: An XML comment starts with <!– and ends with –>. The syntax to include an XML comment is <!– comment statement –> and it can appear anywhere in the document.

Ans. to Q. No. 10: The biggest advantage if an XML tag is that it can be defined on its own based on one’s requirement.

Ans. to Q. No. 11: Yes, XML tags are case sensitive.

Ans. to Q. No. 12: An attribute is a property which can define an element in detail. An attribute can hold only one value at a time.

Ans. to Q. No. 13: Some of the predefined entities in XML are: < (<), > (>), & (&), " (“), ' (‘).

Ans. to Q. No. 14: An XML document is said to **valid** if and only if it is well-formed and it also conforms to the rules specified in the DTD.

Ans. to Q. No. 15: PCDATA means parsed character data which is the text found between the start tag and the end tag of an XML element.



11.13 MODEL QUESTIONS

Q.1: What are the limitations of an HTML document?

Q.2: What is XML? Why is it needed?

Q.3: Explain the uses of XML.

Q.4: Differentiate between HTML and XML.

Q.5: What are the criteria to have a well formed XML document?

Q.6: Why a closing tag for XML declaration is not needed?

Q.7: What are entity references in XML? Explain in Detail.

Q.8: What is Document Type Definition (DTD)? Why is it needed? Explain about it in detail.

Q.9: When can an XML document be said to be valid?

Q.10: Write a short note on XML elements and its attributes.

Q.11: Design a well formed and valid XML document depicting the overall structure of any book.

*** ***** ***

UNIT 12: AJAX BASICS

UNIT STRUCTURE

12.1 Learning Objectives

12.2 Introduction

12.3 AJAX

12.3.1 Technologies in AJAX

12.3.2 Standard AJAX Application

12.4 XML http Request Object

12.4.1 Syntax for Creating XMLHttpRequest Object

12.5 AJAX Request

12.6 AJAX Response

12.7 AJAX Events

12.7.1 Local Events

12.7.2 Global Events

12.8 Let Us Sum Up

12.9 Further Reading

12.10 Answers to Check Your Progress

12.11 Model Questions

12.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define AJAX
- create XML http Request Object
- describe AJAX Request
- describe AJAX Response
- explain AJAX Events.

12.2 INTRODUCTION

Different objects in Javascript like Window, Document, Form Objects, TextBox, TextArea, Button, CheckBox, Select Objects, Date, Match, String Objects, Regular Expressions and Arrays have been covered in detail in the previous units. We have also learnt the basics of XML.

In this unit you will be introduced to AJAX. We will learn about creating XML http Request Object, AJAX Request, AJAX Response and AJAX Events.

In the next unit we will be introduced to PHP a server side programming language. We will learn basic programming concepts of PHP and also learn about variables, data-types, constants, scope of variables, types of variables, type casting, operators, operator precedence and preferences in PHP.

12.3 AJAX

Ajax stands for **Asynchronous JavaScript** and **XML**. It is a technique by which we request data from remote server without reloading the entire page.

Ajax is not a programming language. Rather Ajax is a technique used by web developers in order to make websites behave like desktop applications.

12.3.1 Technologies in AJAX

- HTML – used at client side
- CSS – used at client side
- JavaScript – used to make requests
- XML – are request formats
- JSON – are request formats
- PHP – used at server side

What is Asynchronous in AJAX?

Asynchronous means that we are exchanging data to/from the server in the background without having to refresh the page again and again.

12.3.2 Standard AJAX Application

In standard client server applications, clients have to wait for the response from the server to execute other tasks on the webpage. These are static applications and not dynamic applications.

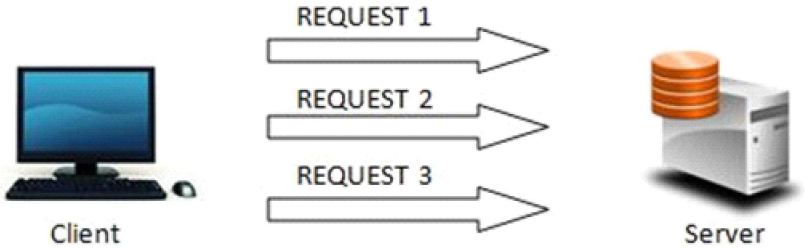


Figure 12.1: Standard AJAX Applications

In standard AJAX applications, clients can make multiple requests concurrently in the same time and doesn't have to wait for a response from the server in order to make a new request. This is a major advantage of AJAX by which we can exchange data from the server or we can post data to the server without having to reload the page again and again.

What about the response in AJAX applications?

In AJAX, responses from the server are handled in the form of callbacks.

A callback is a special function which is used in AJAX so that server can respond to the client when it is ready to send data to the client. So server will say to the client that it is ready and it is just executing this callback function.



CHECK YOUR PROGRESS

Q.1: Name a client side and a server side technology.

.....

Q.2: What is a difference between a standard client server application and a standard AJAX application?

.....

.....

.....

.....

12.4 XML HTTP REQUEST OBJECT

All modern browsers support the XMLHttpRequest object. The XMLHttpRequest object can be used to exchange data with a server behind

the scenes. This means that it is possible to update parts of a webpage, without reloading the whole webpage.

12.4.1 Syntax for Creating XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

variable = new XMLHttpRequest();

Firstly, we create a variable “xhttp” and initialize it to the XMLHttpRequest class.

var xhttp = new XMLHttpRequest();

By this we can declare a simple variable of XMLHttpRequest object.

12.5 AJAX REQUEST

To send a request to the server, we use the open() and send() methods of the XMLHttpRequest object.

xhttp.open(“GET”, “file.txt”, true);

The first parameter in the open() method is the method used in the request (GET or POST). The second parameter is the url of the file that we want to load into our webpage. The third parameter when set to true signifies that we are making asynchronous request and if we set it to false then it will not be an AJAX request.

xhttp.send();

xhttp.send(string);

The send() method is used to send the GET request and send(string) method is used to send the POST request (variable string being the user input) to the server.

12.6 AJAX RESPONSE

When a request is sent to the server, a response will be sent back by the server. This response is to be handled by the client.

There is property of the request XMLHttpRequest that we have made. The property is named as onreadystatechange property. The readyState property holds the status of the XMLHttpRequest. The onreadystatechange property defines a function to be executed when the readyState changes. The status property and the statusText property holds the status of the XMLHttpRequest object.

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK"403: "Forbidden"404: "Page not found"
statusText	Returns the status-text (e.g., "OK" or "Not Found")



CHECK YOUR PROGRESS

Q.3: Name two modern browsers having a built-in XMLHttpRequest object.

.....

Q.4: What is the syntax for creating an XMLHttpRequest object?

.....

.....

Q.5: What are the methods used for sending a request to the server?

.....

Q.6: What property holds the status of the XMLHttpRequest?

.....

.....

12.7 AJAX EVENTS

Ajax requests produces or fires a number of different events that we can subscribe to.

There are two types of events:

1) Local Events
2) Global Events

12.7.1 Local Events

These are callbacks that we can subscribe to within the Ajax request object.

Local Events	Description
beforeSend	This event, which is triggered before an Ajax request is started, allows you to modify the XMLHttpRequest object (setting additional headers, if need be.)
success	This event is only called if the request was successful (no errors from the server, no errors with the data).
error	This event is only called if an error occurred with the request (you can never have both an error and a success callback with a request).
complete	This event is called regardless of if the request was successful, or not. You will always receive a complete callback, even for synchronous requests.

Example 12.1: Script using local events

```
$.ajax({
    beforeSend: function(){
        // Handle the beforeSend event
    },
    complete: function(){
    // Handle the complete event
    }
    // .....
});
```

12.7.2 Global Events

These events are triggered on the document, calling any handlers which may be listening.

Global Events	Description
ajaxStart	This event is triggered if an Ajax request is started and no other Ajax requests are currently running.
ajaxStop	This global event is triggered if there are no more Ajax requests being processed.
ajaxSend	This global event is also triggered before the request is run.
ajaxSuccess	This event is also only called if the request was successful.
ajaxError	This global event behaves the same as the local error event.
ajaxComplete	This event behaves the same as the complete event and will be triggered every time an Ajax request finishes.

Example 12.2: Script using global event

```
$(document).bind("ajaxSend", function(){
        $("#loading").show();
        }).bind("ajaxComplete", function(){
        $("#loading").hide();
});
```

Global events can be disabled for a particular Ajax request by passing in the global option.

```
$.ajax({
        url: "test.html",
        global: false,
        // ...
});
```

Below given is the list of Ajax events in the order in which they are triggered. The indented events are triggered for each and every Ajax request (unless a global option has been set).

- ajaxStart (Global Event)
  - beforeSend (Local Event)
  - ajaxSend (Global Event)
  - success (Local Event)
  - ajaxSuccess (Global Event)
  - error (Local Event)
  - ajaxError (Global Event)
  - complete (Local Event)
  - ajaxComplete (Global Event)
  - ajaxStop (Global Event)



CHECK YOUR PROGRESS

Q.7: What are the two types of AJAX Events?

.....

Q.8: Give two examples each of the two types of AJAX Events.

.....

.....



12.8 LET US SUM UP

- Ajax stands for Asynchronous JavaScript and XML.
- Ajax is a technique by which we request data from remote server without reloading the entire page.
- Ajax is a technique used by web developers in order to make websites behave like desktop applications.
- Standard AJAX applications, clients can make multiple requests concurrently in the same time and doesn't have to wait for a response from the server in order to make a new request.
- Using AJAX, we can exchange data from the server or we can post data to the server without having to reload the page again and again.
- In AJAX, responses from the server are handled in the form of callbacks.
- A callback is a special function which is used in AJAX so that server can respond to the client when it is ready to send data to the client.

- XMLHttpRequest object can be used to exchange data with a server behind the scenes.
- All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.
- To send a request to the server, we use the open() and send() methods of the XMLHttpRequest object.
- When a request is send to the server, a response will be send back by the server which is handled by the client.
- The readyState property holds the status of the XMLHttpRequest.
- The onreadystatechange property defines a function to be executed when the readyState changes.
- The status property and the statusText property holds the status of the XMLHttpRequest object.
- There are two types of events: Local Events and Global Events.
- Local events are callbacks that we can subscribe to within the Ajax request object.
- Global events are triggered on the document, calling any handlers which may be listening.
- Global events can be disabled for a particular Ajax request by passing in the global option.



12.9 FURTHER READING

1) Holzner, Steve. (2006). *Ajax For Dummies*.
2) Riordan, Rebecca M. (2008). *Head First Ajax: A Brain-Friendly Guide*. O'Reilly Media.
3) Zakas, Nicholas C. McPeak, Jeremy & Fawcett, Joe. (2007). *Professional Ajax*. Wrox.



12.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: Client Side – HTML, Server Side – PHP

Ans. to Q. No. 2: In standard client server applications, clients have to wait for the response from the server to execute other tasks on the

webpage. Whereas, in standard AJAX applications, clients can make multiple requests concurrently in the same time and doesn't have to wait for a response from the server in order to make a new request.

Ans. to Q. No. 3: Chrome, Firefox

Ans. to Q. No. 4: Syntax for creating an XMLHttpRequest object:

variable = new XMLHttpRequest();

Ans. to Q. No. 5: Methods open() and send().

Ans. to Q. No. 6: readyState property holds the status of the XMLHttpRequest.

Ans. to Q. No. 7: Local Events and Global Events.

Ans. to Q. No. 8: Local Events – beforeSend, success

Global Events – ajaxStart, ajaxStop



12.11 MODEL QUESTIONS

Q.1: What is AJAX?

Q.2: What are some technologies in AJAX?

Q.3: What is Asynchronous in AJAX?

Q.4: What are standard AJAX applications?

Q.5: What are call backs in AJAX?

Q.6: What is an XMLHttpRequest object?

Q.7: What is the syntax for creating XMLHttpRequest object?

Q.8: What methods of the XMLHttpRequest object are used to send a request to the server?

Q.9: Describe the parameters used in the open() method.

Q.10: What type of request is used to send using send() method and send(string) method?

Q.11: What are the different properties in an AJAX response?

Q.12: What are the types of AJAX events?

Q.13: What are Local Events?

Q.14: What are Global Events?

Q.15: Give examples of Local Events and Global Events.

*** ***** ***

UNIT 13: INTRODUCTION TO PHP

UNIT STRUCTURE

13.1 Learning Objectives

13.2 Introduction

13.3 Getting started with PHP

13.4 A Sample PHP Program

13.5 Variables in PHP

13.6 Data Types in PHP

13.7 Typecasting in PHP

13.8 Types and Scope of Variables in PHP

- 13.8.1 Local Variable
- 13.8.2 Global Variable
- 13.8.3 Static Variable

13.9 Constants in PHP

13.10 Operators in PHP

- 13.10.1 Arithmetic Operator
- 13.10.2 Assignment Operator
- 13.10.3 Comparison Operator
- 13.10.4 Logical or Relational Operator
- 13.10.5 Conditional Operator

13.11 Operator Precedence and Associativity in PHP

13.12 Let Us Sum Up

13.13 Further Reading

13.14 Answers to Check Your Progress

13.15 Model Questions

13.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn the concept of server side scripting
- learn the basic concept of PHP
- describe the different data-types, variables, constants used in PHP language

- learn about scope of variables and typecasting in PHP
- learn different operators and their precedence in PHP.

13.2 INTRODUCTION

Web Scripting is broadly classified in two categories. Client Side Scripting and Server Side Scripting. In Client Side Scripting, the scripts run on the web browser where the source code is transferred from the web server to the user's computer over the internet and run directly in the browser. In Server Side Scripting, the user's request is fulfilled by running a script directly on the web server to generate static content which is then sent to the client computer.

We have already acquainted you with HTML, XML, JavaScript and Ajax in our previous units. We learnt the basics of Client-Side scripting language.

In this unit, you will learn the basics of server-side scripting language PHP.

13.3 GETTING STARTED WITH PHP

PHP stands for *Hypertext Pre-processor* and it is a widely-used, open source server-side scripting language. It is a general purpose scripting language which can be used in websites and web applications for making dynamic and interactive web pages. PHP was originally developed by Rasmus Lerdorf in 1995. Unlike JavaScript, which is a script that runs in a web browser, all PHP code is executed on the web server.

As mentioned, PHP is a general purpose scripting language, which means it can be installed in any computer and can be executed from command line. However, the main purpose of PHP is to compose Dynamic Web Pages which are stored and executed in a web server. PHP is a C-like language therefore you should find most of these syntax familiar. But PHP is different from C/C++ in some scenarios, it will be pointed out specifically as we progress in this lesson.

The PHP files have the extension "*.php*" and can contain text, HTML, CSS, JavaScript, and PHP code. The PHP code that are present in the .php

file is identified by the delimiters “<?php” and “?>”. Whenever the web server encounters a .php file, it finds and execute only those snippets of code which are inside these delimiters, the rest of the code is assumed to be HTML or Plain Text. Please note that, apart from the aforementioned pair of delimiters, PHP understands various other delimiters like “<?” and “?>”; “<script language=”php”>” and “</script>”; “<%” and “%>” etc. However, “<?php” and “?>” is the most widely used and standard delimiters which is used by most of the developers and will be used as convention throughout this section.

In this unit we will learn about basic programming concepts of PHP such as different data-types, variables, constants, operators, Type Casting etc. used in PHP. In the next unit we will explore about Control statements used in PHP where we will learn about uses of if statements, switch statements, various looping statements and the concepts of Arrays. So let’s get started!!

13.4 A SAMPLE PHP PROGRAM

As we always do, let us start with simple “Hello World” program. Although it is very rudimentary, it performs two non-trivial tasks –

1) To check whether the development environment is successfully setup or not and,
2) To get started with the language syntax.

To run PHP in our computer system, we need to setup a local development web server. There are various free and open-source Apache-MySQL-PHP(AMP) packages like WAMP(for Windows), LAMP(for Linux), MAMP(for Macintosh), XAMPP(Cross Platform) etc. We can download the appropriate package for our system from the Internet and get started instantly.

PHP code can be written using any text editor like NotePad, GEdit, TextEdit etc. We have to store the file with extension .php inside our local web server directory (e.g., ‘htdocs’ for ‘XAMPP’, ‘www’ for ‘WAMP’ etc.). It is always a good practice use lowercase characters in filename and to replace white spaces with underscore (‘_’).

Example 13.1: The first “Hello World” Program in PHP

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

The PHP code in the above example is denoted with “<?php” and “?>”. “echo()” is a function that is used in PHP to show output. As demonstrated in the example, “echo” is a language construct and can invoke without using the parenthesis. “echo” takes a string or collection of strings as parameter and shows them as output. Its return type is “void”. “print()”, “printf()” or “sprintf()” functions can also be used alternatively to show output in PHP. Similar to other C-like languages, all statements in PHP must be terminated with a semicolon (;).

13.5 VARIABLES IN PHP

From Computer Programming point of view, a variable is a user defined *keyword* or phrase that uniquely identifies a particular memory location in the system. Unlike C/C++, which is a strictly typed language, PHP is loosely typed language i.e., the data type of the variable doesn’t need to be defined explicitly, the type of the variable is decided depending on the data it stores.

Variables are defined in PHP using the symbol ‘$’ followed by the name of the variable. The variable name can contain letters, numbers and underscore but name of the variable must start with a letter.

Example 13.2: Example showing variables in PHP

```
<?php
      $v = "Hello World";
      echo $v;
?>
```

13.6 DATA TYPES IN PHP

As mentioned in the previous section, PHP is a loosely typed language. So the programmer doesn’t have to define the type of the variable

explicitly. The language decides the type depending on the data stored in the variable. The variables in PHP can be of one of the following types–

1) **Integer:** Integer assumes numeric non-fraction values i.e. Integer can store only whole numbers like 123, 5, 0, -3 etc. In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 59;
var_dump($x);
?>
```

The above code will give output as: int(59)

2) **Float:** Float can store any real numbers. The values are stored in decimal format like 123.45, 5.0, -3.0 etc. In the following example $x is a float variable.

```
<?php
$x = 59.7;
var_dump($x);
?>
```

The above code will give output as: float(59)

3) **Boolean:** It can store only one of the following two values – TRUE or FALSE. Used mainly with logical decisions. In the following example $x is a Boolean variable.

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

The above code will give output as: bool (true)

4) **Strings:** String is a collection of alphanumeric characters like “Hello World”, “H3LL0” etc. In the following example $x is a string variable.

```
<?php
$x = “Hello World”;
Echo $x;
?>
```

The above code will give output as: “Hello World”

5) **Arrays:** It is a collection of homogeneous data types. An array can contain a collection of integers or float values or strings but values should of same data type. E.g., 1, 10, 20, 5, 6000 all these values can be stored in an array of Integers. In the following example $x is a array variable.

```
<?php
$x = array("red","green","blue");
echo "$x[0] <br> $x[1] <br> $x[2]";
?>
```

The above code will give output as:

red
green
blue

6) **Objects:** An object is a data type which stores data and information on how to process that data. An object is a specific instance of a class which serve as templates for objects. Objects are created based on this template via the new keyword. In the following example $maruti is an object of class car.

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

The above code will give output as: VW

7) **NULL:** 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. A variable created without a value, is automatically assigned a value of NULL. In the following example $x is a NULL variable.

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

The above code will give output as: NULL NULL

8) **Resources:** A resource is a special variable, holding a reference to an external resource such as database connections, file opening etc.

13.7 TYPE CASTING

Type Casting, also known as type conversion, is a method in computer programming using which the value in one data type can be converted into another data type. In most C-like languages, there exist two types of type casting– one is *implicit* type casting and the other is *explicit*.

Implicit type casting is performed by the compiler/interpreter itself without requiring any intervention from programmer's end. It is generally performed to mitigate the chance of data loss by upgrading data type of lower domain to data type of higher domain where there exists more than one data type in an expression. For example,

Example 13.3: Implicit Type Casting

```
<?php
      $i = 5;
      $j = 0.5;
      echo $i + $j;
?>
```

The output of the above code will be 5.5. Here $i is an integer variable as it contains a whole number 5 and $j is a float variable. As float has larger

domain than integer therefore the integer variable $i is implicitly casted to float to avoid data loss. Thus value of $i is casted to **5.0**.

Explicit type casting is defined by the programmer to convert values in one data type to another explicitly. The explicit type casting mechanism is PHP is identical to C/C++. The programmer only needs to mention the type of the resultant data type within parenthesis and the value will be converted to that mentioned data type.

```
<?php
      $i = 5.5;
      $j = (int) $i;
      echo $j;
?>
```

Example 13.4: Explicit Type Casting

In Example 13.4, we've defined a float type PHP variable $i containing the value 5.5. The value is casted to Integer using "(int)" and stored at $j. So $j is now an integer type variable containing the value 5 and decimal part of $i i.e. ".5" is lost in the process of conversion.

Please note that, type casting from String to any numeric data type like Integer or Float always results in 0 unless the string starts with a number. If the string starts with a number, that number is returned after conversion.

For example–

```
<?php
      $i = "5 Mangoes";
      $j = "Five Mangoes";
      $k = (int) $i;
      $l = (int) $j;
      echo $k . "<br>" . $l;
?>
```

Example 13.5: Explicit Type Casting in Strings

The above gives the output 5 and 0 respectively. Please note, we can concatenate strings in PHP with dot(.) operator. We're using "<br>" inside in echo() to print the output in separate lines.



EXERCISE 13.1

Write a Program in PHP to define a float variable named "pi" and show its integer part and float part separately in two different lines.

13.8 TYPES AND SCOPE OF VARIABLES IN PHP

Scope of a variable is the range or area of the program in which a variable is visible or available to the program. In another term, it can also be defined as the lifetime of the variable. Depending on the scope, variables in PHP can be classified in three broad categories–

1) Local Variable
2) Global Variable
3) Static Variable

13.8.1 Local Variable

A variable declared and used inside a function only is known as a local variable. Local variables are accessible only within the function and are destroyed as soon as the function completes its execution.

```
<?php
    $x = 10;
    function local_scope() {
        $x = 5;
    echo "Inside Local Scope" . $x. "<br>";
    }
    local_scope();
    echo "Outside Local Scope". $x;
?>
```

Example 13.6: Local Variables in PHP

As we can see in the example, there are two different declaration of variable $x. But they are different. The scope of $x

inside the function local_scope() and it assumes the value 5, whereas the $x outside the function local_scope() assumes the value 10. This $x is also known as global variable as it is not inside any function and can be accessed anywhere inside the program. We will define global variables in the next sub section.

13.8.2 Global Variable

As we've already discussed in the previous sub section, Global variables are declared outside functions and they can be accessed from anywhere inside the program. The global variables exist in the memory throughout the period of execution of the program. Although global variables can be accessed from anywhere in the program, in order to modify it inside a particular function, we've to explicitly declare it as global inside that function with the help of the keyword **GLOBAL**. Let's modify the previous example–

```
<?php
    $x = 10;
    function local_scope() {
        GLOBAL $x;
        $x = 5;
    echo "Inside Local Scope" . $x. "<br>";
    }
    local_scope();
    echo "Outside Local Scope". $x;
?>
```

Example 13.7: Modifying Global Variables in PHP

Unlike the previous example, this time local_scope() modifies the global instance of the variable $x instead of creating another local variable $x because we've explicitly mentioned inside the function to use the global version of $x by applying the keyword **GLOBAL**.

Please note that, PHP stores record of all the global variables present inside the program using an array called $GLOBALS. There

are some predefined global variables present in PHP which exists in the program by default, programmers don't have to explicitly declare them. These are known as Super Globals. The Super Globals are not restricted by any scope. $GLOBALS is one such Super Global. Some other Super Globals present in PHP are $_POST, $_GET, $_FILES, $_COOKIE, $_SESSION, $_REQUEST etc. We'll study about some of these in later portions.

13.8.3 Static Variable

From the definition of local variables we know that lifetime of local variables are limited to the execution period of the function inside which it is declared. As soon as the function completes its execution, the local variables associated with it are removed by the interpreter. But in some cases, we need to retain some of these local variables even after the execution of the function inside which they are declared. We need to declare such variables as static variable. Static variables are defined with the keyword **static**.

13.9 CONSTANTS IN PHP

Constants are those entities in a program whose value doesn't change throughout the execution of the program. Constants are defined in PHP with the function define(). Constant's naming convention is identical to that of variables. Constants in PHP are case-sensitive by default but user can modify this behaviour using a special parameter inside define() function.

define() functions takes three parameters – first parameter is name which defines the name of the constant, second one is values which assigns a value to the constant and third one is the option parameter for declaring whether the constant is case-insensitive or not. By default it is set to false.

```
<?php
define("HELLO", 123);
echo HELLO;
?>
```

Example 13.8: Declaring Constant in PHP

The above example declares a case-sensitive constant HELLO whose value is set to 123.

Another way of declaring a constant was introduced in PHP 5.3 which is with the help of keyword **const.** We can define the constant with **const** keyword followed by the name of the constant. One fundamental difference between const and define() is that const works at compile time whereas define() works at run time.



CHECK YOUR PROGRESS

Q.1: What will be the output of the following PHP code?

```
<?php
echo “echo “Hello World””;
?>
```

a) Hello world b) echo “Hello world”

c) echo Hello world d) Syntax Error

Q.2: What will be the output of the following PHP code?

```
<?php
$x = 5; $y = 10;$z = “$x + $y”; echo “$z”;
?>
```

a) 15 b) 10 + 5 c) $z d) $x + $y

Q.3: What will be the output of the following PHP code:

```
<?php
$x = 10.8;$y = 4;$z = 3;echo ($x % ($y) + $z);
?>
```

a) 5 b) 3 c) 0 d) 1

Q.4: What will be the output of the following PHP code:

```
<?php
$x = 3.5; $y = 2;$z = 2;echo $x / $y / $z;
?>
```

a) 1.75 b) 0.875 c) 3.5 d) 1

Q.5: What will be the output of the following PHP code:

```
<?php
define("hello", "my first PHP code");
echo $hello;
?>
```

a) $hello b) no output

c) my first PHP code d) hello

Q.6: What will be the output of the following PHP code:

```
<?php
GLOBAL $x;
$x = 10;
function local_scope() {
$x = 5;
echo "$x t";
}
local_scope();
echo $x;
?>
```

a) 5 10 b) 10 5 c) 5 5 d) error

13.10 OPERATORS IN PHP

Operators are certain elements present in a programming language which take some expressions known as operands and perform some pre-defined actions on them to generate some results. Operators perform various tasks like assigning values, performing arithmetic and logical operations, manipulating strings etc.

Operators are broadly classified into following categories –

1) Arithmetic Operators
2) Assignment Operators
3) Comparison Operators
4) Logical or Relational Operators
5) Conditional Operator

13.10.1 Arithmetic Operators

Arithmetic operators are used to perform some arithmetic operations in the program. Arithmetic operators can be both binary and unary. Arithmetic operators present in PHP are as follows–

Table 13.1: List of Arithmetic Operators

OPERATOR	DESCRIPTION	TYPE
+	Addition – Adds two operands	Binary
–	Subtraction – Subtracts the second operator from the first	Binary
*	Multiplication – Perform multiplication of two operators	Binary
/	Division – Divides the first operator by the second	Binary
%	Modulus – Finds the reminder after performing integer division	Binary
++	Increment – Increases the value of the operand by 1	Unary
—	Decrement – Decreases the value of the operand by 1	Unary

The following code shows the use of different arithmetic operators in PHP:

```
<?php
      $a = 42;
      $b = 20;

      $c = $a + $b;
      echo "Addtion: $c <br/>";

      $c = $a - $b;
      echo "Substraction: $c <br/>";

      $c = $a * $b;
      echo "Multiplication: $c <br/>";
```

```
        $c = $a / $b;
        echo "Division: $c <br/>";

        $c = $a % $b;
        echo "Modulus: $c <br/>";

        $c = $a++;
        echo "Increment: $c <br/>";

        $c = $a–;
        echo "Decrement: $c <br/>";
?>
```

The above program will produce the result:

```
Addtion:        62
Substraction:   22
Multiplication: 840
Division:       2.1
Modulus:        2
Increment:      42
Decrement:      43
```

13.10.2 Assignment Operators

Assignment operators are used to assign results of some operations. "=" is the most commonly used assignment operators which we have already used in most of our examples. Assignment Operators present in PHP are–

Table 13.2: List of Assignment Operators

OPERATOR	DESCRIPTION	TYPE
=	Simple assignment operator, Assigns values from right side operands to left side operand	Binary
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	Binary

-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	Binary
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	Binary
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	Binary
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	Binary

The following code shows the use of different assignment operators:

```
<?php
      $a = 12;
      $b = 20;

      $c = $a + $b;
      echo "Addtion Operation Result: $c <br/>";

      $c += $a;
      echo "Add AND Assigment Operation Result:
      $c <br/>";

      $c -= $a;
      echo "Subtract AND Assignment Operation
      Result: $c <br/>";

      $c *= $a;
      echo "Multiply AND Assignment Operation
      Result: $c <br/>";

      $c /= $a;
      echo "Division AND Assignment Operation
      Result: $c <br/>";
```

```
        $c %= $a;
        echo "Modulus AND Assignment Operation
        Result: $c <br/>";
?>
```

The above program will produce the result:

```
Addtion Operation Result:                    62
Add AND Assigment Operation Result:          104
Subtract AND Assignment Operation Result:    62
Multiply AND Assignment Operation Result:    2604
Division AND Assignment Operation Result:    62
Modulus AND Assignment Operation Result:     20
```

13.10.3 Comparison Operator

Comparison operators are used in PHP to compare two values against some criteria like whether they are equal or not, whether first one is greater than the second one etc. Comparison operators return Boolean values and used mostly with conditional statements like if, if-else etc. Comparison operators present in PHP are–

Table 13.3: List of Comparison Operators

OPERATOR	DESCRIPTION	TYPE
==	Checks if the value of two operands are equal or not.	Binary
!=	Checks whether two operands are unequal or not.	Binary
>	Checks if the value of left operand is greater than the value of right operand.	Binary
<	Checks if the value of left operand is less than the value of right operand.	Binary
>=	Checks if the value of left operand is greater than or equal to the value of right operand	Binary
<=	Checks if the value of left operand is less than or equal to the value of right operand.	Binary

The following code shows the use of comparison operator “>”:

```
<?php
   $a = 12;
   $b = 20;
   if( $a > $b ) {
      echo “TEST2 : a is greater than  b<br/>”;
     }
   else {
      echo “TEST2 : a is not greater than b<br/>”;
   }
?>
```

The above program will produce the result:

TEST2 : a is not greater than b

13.10.4 Logical or Relational Operators

These operators are used to combine conditional statements. In other words, these operators try to establish a relationship among various conditions. Similar to comparison operators, they return Boolean values. Conditional operators present in PHP are–

Table 13.4: Logical Operators in PHP

OPERATOR	DESCRIPTION	TYPE
and	Logical AND operator. If both the operands are true then condition becomes true.	Binary
or	Logical OR Operator. If any of the two operands are true then condition becomes true.	Binary
xor	Logical XOR Operator. If any of the two operands are true then condition becomes false otherwise true.	Binary
&&	Logical AND operator. If both the operands are true then condition becomes true.	Binary
||	Logical OR Operator. If any of the two operands are true then condition becomes true.	Binary

!	Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then it will return false and vice versa.	Binary

The following code shows the use of logical operators in PHP:

```
<?php
$year = 2014;
// Leap years are divisible by 400 or by 4 but
not 100
if(($year % 400 == 0) || (($year % 100 != 0) &&
($year % 4 == 0))){
    echo "$year is a leap year.";
} else{
    echo "$year is not a leap year.";
}
?>
```

The above program will produce the result:

2014 is not a leap year.

13.9.5 Conditional Operator

This is a ternary operator which is used to make some logical decision. It consist of three operands. First operand evaluates to a Boolean values. If the result of first operand is true then second operand is executed otherwise third operand gets executed. It can be visualized as a shorthand of if-else statement which we will study in the next section.

Operator	Description	Type
? :	Conditional Expression	Ternary

Table 13.5: Conditional Operators in PHP

```
<?php
    $x = 5;
    $y = ($x > 5) ? "X > 5" : "X <= 5";
    echo $y;
?>
```

Example 13.9: Using Conditional Operator in PHP

The above program shows the output “X <= 5” as the condition $x > 5 becomes false and the third operand after: is returned and stored in $y.



Exercise 13.2

Write a Program in PHP to define an integer variable “num”. Check whether the “num” is divisible by 7 or not. If “num” is divisible by 7 then square of the “num” should be displayed otherwise return the reminder obtained.

13.11 OPERATOR PRECEDENCE AND ASSOCIATIVITY IN PHP

Operator precedence and associativity are some criteria which are taken into consideration while evaluating a complex expression consisting of various operators. It is used in both computer programming and mathematics.

Precedence is a criteria which is checked to decide which operator to evaluate first from a series of operators present in a complex mathematical expression. Essentially, precedence decides the order in which operators are evaluated. Associativity of the operators is required when the operators of equal precedence is encountered in an expression.

Below is a list of PHP operators along with their associativity. The list is compiled in decreasing order of precedence i.e. operators with higher precedence are on top. Operators with equal precedence are grouped together.

ASSOCIATIVITY	OPERATORS
non-associative	*clone new*
left	*[*
right	**
right	++ — ~ (int) (float) (string) (array) (object) (bool) @
non-associative	instanceof

right	!
left	* / %
left	+ - .
left	<< >>
non-associative	< <= > >=
non-associative	== != === !== <> <=>
left	&
left	^
left	|
left	&&
left	||
right	??
left	? :
right	= += -= *= **= /= .= %= &= |= ^= <<= >>=
left	and
left	xor
left	or

Table 6: PHP Operator Precedence. *Courtesy: php.net*



CHECK YOUR PROGRESS

Q.7: What will be the output of the following PHP code?

```
<?php
$a = 10;
echo ++$a;
echo $a++;
echo $a;
echo ++$a;
?>
```

a) 11111213 b) 11121213

c) 11111212 d) 11111112

Q.8: What will be the output of the following PHP code?

```
<?php
$x = 1;
$y = 2;
if (++$x == $y++)
{
        echo "true", $y, $x;
}
?>
```

a) true 32 b) true 23 c) true 22 d) true 33

Q.9: What will be the output of the following PHP code:

```
<?php
$y = 2;
if (—$y == 2 || $y xor —$y)
{
        echo $y;
}
?>
```

a) 1 b) 2 c) 0 d) no output

Q.10: What will be the output of the following PHP code:

```
<?php
$i = 1;
if ($i++ && ($i == 1))
        printf(“Yesn$i”);
else
        printf(“Non$i”);
?>
```

a) No 2 b) Yes 1 c) Yes 2 d) No 1

Q.11: What will be the output of the following PHP code:

```
<?php
$b = 1; $c = 4; $a = 5;
$d = $b + $c == $a;
        print $d;
?>
```

a) 5 b) 0 c) 10 d) 1

Q.12: What will be the output of the following PHP code:

```
<?php
$i = 0;
$x = $i++; $y = ++$i;
print $x; print $y;
?>
```

a) 01 b) 02 c) 12 d) 21



13.12 LET US SUM UP

- PHP stands for Hypertext Pre-processor and it is a widely-used, open source server-side scripting language.
- In Server Side Scripting, the user's request is fulfilled by running a script directly on the web server to generate static content which is then sent to the client computer.
- The PHP files have the extension "*.php*" and can contain text, HTML, CSS, JavaScript, and PHP code.
- Variables are defined in PHP using the symbol '$' followed by the name of the variable.
- The data types in PHP are integer, float, Boolean, string and array
- Depending on the scope the variables in PHP can be classified in three broad categories – *local Variable*, *global Variable* and *Static Variable*
- Constants are defined in PHP with the function define().
- Operators are broadly classified into following categories – *Arithmetic Operators, Assignment Operators, Comparison Operators, Logical or Relational Operators* and *Conditional Operator.*



13.13 FURTHER READING

1) Julie C. M., Sams. *Teach Yourself PHP in 24 Hours, Techmedia*.
2) Welling, Luke, Thomson, Laura. *PHP and MySQL Web Development, Addison-Wesley Professional*.
3) http://php.net, Official website of PHP.



13.14 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: (d) **Ans. to Q. No. 2:** (b)
Ans. to Q. No. 3: (a) **Ans. to Q. No. 4:** (b)
Ans. to Q. No. 5: (b) **Ans. to Q. No. 6:** (a).
Ans. to Q. No. 7: (a) **Ans. to Q. No. 8:** (a)
Ans. to Q. No. 9: (c) **Ans. to Q. No. 10:** (a)
Ans. to Q. No. 11: (d) **Ans. to Q. No. 12:** (b)



13.15 MODEL QUESTIONS

Q.1: Write down the differences between Client Side and Server Side Scripting in Web Development.

Q.2: What are delimiters that can be used to signify PHP code in a file?

Q.3: What is a variable? What are the different types of variables present in PHP? Give proper programming example.

Q.4: What is data type? What are the different types of data types present in PHP?

Q.5: Define type conversion? Why is it required? Differentiate between implicit and explicit type conversion.

Q.6: Write PHP code to define a string variable and display it in bold font using HTML. Also type-cast it to an integer and show the output.

Q.7: What are the different ways of defining a constant in PHP? Explain with proper syntax.

Q.8: Differentiate between define() and const.

Q.9: What are the various types of arithmetic operators present in PHP? What is the usage of modulus operator? Give a programming example.

Q.10: Define two integer variables in PHP and perform all the arithmetic operation in these two integers and show the result using printf().

Q.11: What is the difference between unary and binary operators?

Q.12: Why do we need static variables in a program?

Q.13: Define associativity and precedence in PHP.

Q.14: Give examples of two server side scripting languages.

*** ***** ***

UNIT 14: CONTROL STATEMENTS IN PHP

UNIT STRUCTURE

14.1 Learning Objectives

14.2 Introduction

14.3 Branching in PHP

14.4 if-else Statement in PHP

14.5 if-elseif-else Statement in PHP

14.5.1 Nested Conditionals

14.6 Switch Statement in PHP

14.6.1 Limitations of Switch

14.7 Looping in PHP

14.7.1 For loop

14.7.2 While loop

14.7.3 Do-While loop

14.7.4 Foreach loop

14.8 Let Us Sum Up

14.9 Further Reading

14.10 Answers to Check Your Progress

14.11 Model Questions

14.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn the branching and looping statements in PHP
- work with various branching and looping techniques
- make use of nested conditionals
- make use of Logical and Relational operators for decision making
- learn the concept of array in PHP.

14.2 INTRODUCTION

In the previous unit, we have discussed the fundamental programming structure of PHP. We have also discussed the various programming constructs like variables, constants, operators etc.

Now, in this unit, we will study the various control structures present in PHP. PHP has different branching and looping mechanisms to facilitate various use cases. Branching, as the name suggests, is used for creating some execution paths in the program which are executed depending on whether some certain conditions occur or not. That's why they are also known as conditional statements.

Looping refers to the process of executing some common tasks/ statements repeatedly unless and until some certain condition gets satisfied. Looping helps programmers to automate some certain repetitive tasks.

14.3 BRANCHING IN PHP

Branching or conditionals or conditional statements provide the program the capability to make some logical decision on its own and to perform some specific tasks (execute a block of code) depending on whether a certain condition is satisfied or not. Branching is a very important concept in case of programming because it makes the program suitable for various use cases.

Let us take a real life scenario to understand the usage of branching. Suppose, we are writing a program which allows the users to choose a candidate. However, for voting there are some criteria that is to be satisfied. For example, in India for voting, a voter must be at least 18 years old. So our program should query the user his/her age and if the age is greater than or equal to 18, he/she is allowed to vote. Else, an appropriate message should be given. Such kind of decision making can be implemented using conditional.

So, in order to implement it in PHP code, let us put emphasis on the phrase *"If the age is greater than or equal to 18, he/she is allowed to vote. Else, an appropriate message should be given".* We can directly translate it to working PHP code using "if-else" conditionals present in the language and in the next section we'll do exactly that.

14.4 if-else STATEMENT IN PHP

The if-else statement is used to make binary decision i.e., if the decision is true, one block of code is executed otherwise if the condition is

false, another block of code is executed. Therefore, if-else breaks or branches the program into two different execution paths. Let us see its syntax:

Syntax of if-else:

```
if(condition)
{
    /*Statements to be executed
    if condition is true*/
}
else
{
    /*Statements to be executed if
    condition is false*/
}
```

From the syntax, it can be inferred that “condition” present inside “if” requires a Boolean return type. If we recall about PHP operators we have studied in the previous unit, we will find that both Logical and Relational operators return values in Boolean. Hence, they come quite handy in the field of decision making.

Commenting in PHP is identical to C/C++. “//” is used to denote single line comment and “/*” and “*/” is used to denote multiple line comment. We have used “/*” and “*/” to reflect the fact that code blocks can have multiple statements.

Let us go back to our problem at hand i.e., the voting machine code. Let us re-visit our problem statement again– “*If the age is greater than or equal to 18, he/she is allowed to vote. Else, an appropriate message should be given”.* From this we can clearly see that it is a binary problem – voter is at least 18 or not. So it can be translated into valid PHP code as

```
$age >= 18
```

where, $age is an integer variable holding the age of the voter. So, we have solved our condition part. Let us see our respective code blocks – if $age >= 18, a message “Great!! You can vote” is displayed; otherwise we

will show “Oops!! You’ve to wait till you’re 18". So, the resultant code of our problem statement can be give as

Example 14.1: Implementing Voting Criteria with if-else

```
<?php
    $age = 43;
    if($age >= 18) {
     echo “Great!! You can vote”;
    }
    else {
     echo “Oops!! You have to wait till you’re 18";
    }
?>
```

Please note that, we can avoid the curly braces after “if” and “else” as they contain only one statement. However, for the sake of having an identical structure with the syntax, the curly braces are used. We have initialized the value of $age to 43 using assignment operator “=” but in real world scenario there must be an option for users to input their age. User inputs in PHP can be taken using HTML form elements. We will study in detail about that in our next unit.



EXERCISE 14.1

Re-write the code given in *Example14.1* to show the remaining age of person to be a voter if he/she is under aged. For e.g., if the age given is 11, then the system should show “Oops!! You’ve to wait 7 years for voting”. Hint – dot(.) operator in echo and Arithmetic Operators.

Another important point to remember is that we can omit the else part of if-else. The above code can be written without else part but the problem is that if the voter’s age is below 18, he/she would not get any output from the system.

14.5 if-elseif- else STATEMENT IN PHP

We have implemented our original problem statement of voting machine using “if-else”. Let us take it a step further. What if we want to show a different congratulatory message to a user who has just turned 18 something like “Congrats!! Welcome to Voting”. Something like that cannot be implemented using “if-else” only because they can iterate over only one condition. To extend the functionality of “if-else” so that it can cater more than one condition, another construct is introduced in between “if” and “else” called “elseif”. “elseif” requires its own condition and there is no restriction on number of “elseif” we can declare in between “if” and “else”. Let us see its syntax:

Syntax of if-elseif-else:

```
if (condition 1) {
    /*Statements to be executed if condition 1
    is true*/ }
elseif (condition 2) {
    /*Statements to be executed if condition 2
    is true*/ }
elseif (condition K) {
    /*Statements to be executed if condition K
    is true*/  }
else {
    /*Statements to be executed if default
    condition is reached*/ }
```

From the syntax, we can see that K number of eleseif can be introduced in between “if’ and “else”, where K is an integer number. So we will try to implement our updated problem statement for voting system using “if-elseif-else”. Let us first visit the updated problem statement– *“If the age is greater than 18, he/she is allowed to vote. Else if, the age is equal to 18, then a congratulatory message should be displayed and should be allowed to vote. Else, an appropriate message should be given.”* This problem statement can be implemented with “if-elseif-else” as–

Example 14.2: Implementing Voting Criteria with if-elseif-else

```
<?php
    $age = 43;
    if($age > 18) {
       echo "Great!! You can vote";
    }
    elseif($age == 18) {
     echo "Congratulations!! Welcome to voting";
    }
    else {
     echo "Oops!! You have to wait till you're 18";
    }
?>
```



EXERCISE 14.2

Implement PHP conditionals for this problem statement– "If the age is greater than 18, he/she is allowed to vote. Else if, the age is equal to 18, then a congratulatory message should be displayed and should be allowed to vote. If the age of the person is above 60, then display a message appreciating the fact that he/she has come to vote. Else, an appropriate message should be given".

14.5.1 Nested Conditionals

It is worth mentioning that we can incorporate more conditions inside the existing if and else blocks or other conditionals. For example, in out voting system problem, if we want to show different messages to boys and girls of 18 years, we need to add another if else block to check their gender after we check their age in the parent **if**. Such technique of adding conditionals inside another conditional is called nesting.

```
if (expression 1 )
{
    if (expression 2 )
    {
        // statements 1
    }
    else
    {
        // Statements 2
    }
    }
    else
    {
    if ( expression 2)
    {
        // Statements 3
    }
    else
    {
        // Statements 4
    }
}
```

14.6 SWITCH STATEMENT IN PHP

On inspecting the syntax of “if-elseif-else”, we can see that it will become tedious to manage all elseif blocks if the number of conditions becomes very high. Suppose, we are writing a program which converts numbers into words. For example, if we give an input as 2, the result is displayed as “Two”. So, if we implement this problem in “if-elseif-else”, we’ll have ten different code blocks for 0-9 and for each code block, a comparison is to be made to check whether the input is equal to that particular digit. It is just a simple example, but in real world scenarios, the situation becomes even worse.

In this kind of scenario, where there are many code blocks and only one of them needs to be executed depending on some criteria, another conditional control named “Switch” is utilized.

Switch is a multi-option branching mechanism where a selection is made over a collection of cases for a particular condition and appropriate code block is executed if a case is satisfied, otherwise **default** code block gets executed. Its syntax is as follows:

```
switch (n){
  case label1:
    //code to be executed if n=label1;
      break;
 case label2:
    //code to be executed if n=label2;
      break;
   case label3:
    //code to be executed if n=label3;
      break;
    ......................
    default:
    //code to be executed
    //if n is different from all labels;
}
```

We can see that, there’s a “break” statement associated with each **case** block. Identical to C/C++, **break** is used to break out of the current branching statement and move on to the succeeding statements. The counterpart of **break** is **continue**. We’ll discuss about both of these in our looping subsection.

The labels (label1, label2, label3 etc.) are some constants against which the switch variable ‘n’ is matched. Please note that we can combine multiple cases into one block of code so that two or more labels corresponds to a single code block.

Let us now try to implement our example using **switch**:

Example 14.3 : switch-case in action

```
<?php
$num = 5;
switch ($num) {
    case 1:
        echo “One”;
        break;
    case 2:
        echo “Two”;
        break;
    case 3:
        echo “Three”;
        break;
    case 4:
        echo “Four”;
        break;
    case 5:
        echo “Five”;
        break;
    case 6:
        echo “Six”;
        break;
    case 7:
        echo “Seven”;
        break;
    case 8:
        echo “Eight”;
        break;
    case 9:
        echo “Nine”;
        break;
```

```
        default:
            echo "Invalid Input!!";
    }
    ?>
```

In the above example, the input is stored in a variable $num (which is inferred as integer variable by PHP interpreter). The input is then checked against the labels of each case. If there's a matching case, the corresponding code block after ':' is executed. Otherwise default statement is echoed out.



EXERCISE 14.3

Extend the above example (Example 3) to work with two digit numbers. For example, if $num variable is initialized to 42, the system should show the output as "Four Two". Hint – Break double digits to single digit, Division by 10, Remainder by 10.

14.6.1 Limitations of Switch

Although **switch** is a very effective conditional control, it has some limitations of its own. One of the major limitations of switch is that it cannot handle logical and relational decisions as opposed to if-else statements. Another drawback is that labels can work with only constants, variable values are not allowed.



CHECK YOUR PROGRESS

Q.1: What will be the output of the following PHP code?

```
<?php
$x;
if ($x)
    print "hello" ;
else
    print "how are you";
?>
```

a) how are you b) hello

c) error d) no output

Q.2: What will be the output of the following PHP code?

```
<?php
$x;
if ($x == 0)
    print “who” ;
else
    print “how are you”;
    print “who”
?>
```

a) how are youwho b) whowho

c) who d) no output

Q.3: What is missing from case 2 in the switch statement below:

```
<?php
$variable = 1;
switch ($variable) {
    case 1:
        echo “Hello World”;
        break;
    case 2:
        echo “Hello”;
    default:
        echo “Hi”;
        break;
    }
?>
```

a) break: b) break();

c) break; d) none of the above

Q.4: What will be displayed in a browser when the following PHP code is executed:

```
<?php
$variable = 2;
```

```
switch ($variable) {
    case 1:
        echo “Hello World”;
        break;
    default:
        echo “Hello You”;
        break;
    }
?>
```

a) Hello World b) Hello You

c) Hello WorldHello You d) error

14.7 LOOPING IN PHP

As we have discussed, at the beginning of this section, the loop, in programming, is a series of statements which are repeatedly executed or iterated for a certain number of times unless some condition is met. Looping is one of those phenomena which makes programmer’s life easy. It saves programmer from writing manual code for repetitive tasks. For example, suppose in a program, the programmer needs to have a list of prime numbers from 1 to 20. Instead of finding and listing all the prime numbersmanually, the programmer can write a loop with few lines of code and the result will be obtained. He/she can now extend the code block to print prime number within any range say 1 to 10000, 500 to 5000 etc. just by changing the value of the **loop variable** and **termination condition**.

The number of iterations a loop will make is dependent upon the termination condition of the loop. However, it may also be possible that a loop may run endlessly due to some logical error at termination condition or the programmer deliberately wants it to run forever. Such kind of loop are termed as **infinite loop**. As a novice programmer, you may accidently end up creating some infinite loops due to logical errors. You can terminate such loop with the keyboard combination Ctrl+C(in case of Windows and Linux CLI). Rest assured, the host OS will eventually terminate the program

which generates infinite loop even if you don't press the aforementioned combination unless you're writing some very low level code.

PHP contains all three standard loop controls present in C/C++ viz. **for**, **while** and **do-while**. Apart from these, PHP also provides another special loop known as **for each** for iterating over an array. Please note that, later versions of C++ also have support for foreach loop. For-each loop can iterate over any type of array, whether it is an array of pre-defined data types like array of integers or float or an array of user defined objects etc. It is powerful and widely utilized PHP loop. We'll study all these four loops along with their syntax and working mechanism.

A loop generally consists of four parts– Initialization of loop variable, loop condition, Increment/Decrement of loop variable and loop code. Loop condition decides when to terminate a loop, Increment or Decrement is used to increase/decrease the loop variable (increment/decrement amount must be linear throughout the loop) and code block that is to be executed repeatedly. Except for-each, the working mechanism of all the other three loops incorporate these components.

14.7.1 For loop

The **for loop** contains the initialization and increment/ decrement of the loop variable and termination condition of the loop in the beginning of the loop syntax followed by the loop code. Its general structure is as follows–

Structure of for loop:

```
for (loop_variable initialization; termination
condition; increment/decrement)
{
    /* loop code */
}
```

Let us consider an example. We have to print all the even number from 1 to 100. So our required series is 2, 4, 6, 8, 10, 12, ……, 100. We can see that the loop starts at 2 and terminates at

100 and distance between each successive value is 2. So we can infer that the *loop_variable* should be initialized to 2, termination condition is – when *loop_variable* becomes 100 and increment value is 2. Having all these details, we can convert the given problem to working PHP code as follows–

Example 14.4: Printing even numbers with for loop

```
<?php
    for($i = 2; $i <= 100; $i = $i+2) {
        echo $i . "<br>";
    }
?>?>
```

Here, $i is the loop variable. As the loop code contain only one line, we can omit the curly braces, however for the sake of better code readability we used them in our code.



EXERCISE 14.4

Write PHP code to print geometric sequence of 3 starting from 9 to 500 (9, 27, 81, 243….) using for loop. Hint –pow() function in PHP.

14.7.2 While loop

In **while loop**, the loop_variable is initialized before the loop declaration, followed by termination condition. Here, the increment/ decrement of the loop_variable is done inside the loop body. The general structure of while loop is as follows–

```
loop_variable initialization;
while (termination condition) {
    /* loop code */
increment/decrement;
}
```

Let's try to implement our previous example of printing even numbers from 1 to 100 using while loop–

```
<?php
$i = 2;
while($i <= 100) {
echo $i . "<br>";
$i = $i+2;
}
?>
```

Example 14.5: Printing even numbers with while loop

Please note that, in this example, we cannot omit the curly braces because there's more than one line inside the loop body.



EXERCISE 14.5

Write PHP code to print 50 random numbers between 10 to 1000 using while loop. Hint – rand() function in PHP.

14.7.3 Do-while loop

Do-while loop is identical to while loop except the fact that here the loop condition occurs after the loop body. There's a peculiar outcome of this – as the loop condition is checked after executing the loop body, the do-while loop runs atleast once even if the loop condition is false. Due to this idiosyncrasy, the do-while loop sometimes become very handy. Let's see its structure–

Structure of do-while loop:

```
loop_variable initialization;
do {
    /* loop code */
    increment/decrement;
} while (termination condition);
```

Following example prints the even numbers between 1 to 100 can be written using do while loop as follows –

Example 14.6 : Printing even numbers with do-while loop

```
<?php
    $i = 2;
    do {
        echo $i . "<br>";
        $i = $i+2;
    } while($i <= 100);
?>
```



EXERCISE 14.6

Write PHP code to print multiplication table of a given number using do-while loop.

14.7.4 Foreach loop

As mentioned earlier, **foreach** loop is utilized in PHP to iterate over an array. It is special kind of loop available in PHP only.

Array in PHP: Before discussing the foreach loop, let us discuss a little bit about array. We know that, array is a collection of similar type of elements. There's no restriction over whether it contains pre-defined or user defined data type but all of them has to be of same data type. There's a striking difference between array in PHP and C/C++. In PHP, the size of the array is not required while declaring the array. The array will incorporate the contents unless memory permits thereby resulting in a dynamic array. Different types of array initialization in PHP –

Example 14.7: Array initialization is PHP

```
<?php
    $a = array("Item1", "Item2", "Item3");
    var_dump($a);
```

```
    $empty_array = array();
    var_dump($empty_array);
    echo count($a);
    echo $a[2];
?>
```

The above example shows the creation of an array of strings named $a. var_dump() is a debugging function that returns information about a variable. It can be used anywhere in PHP code to obtain critical debugging information. The count() function returns the size of the array. Similary array_push() function can be used to insert item at the end of an array.Just like C/C++, we can print an item in array using its index number.

Apart from linear array, PHP supports multidimensional array and associative arrays. Associative arrays are collection of key-value pairs which are bit advanced and out of the scope of this series.

After reviewing a bit about array, let's try to implement the foreach loop on them. foreach loop does not maintains any loop variable explicitly and moves from the beginning of an array item by item. First let's have a look at its structure–

Structure of foreach loop:

```
foreach($array as $variable)
{
    /* loop code */
}
```

Consider the code "$array as $variable", what it does is, it visits the array from the beginning and assigns the current array element is made available to the loop code using the variable $variable. 'as' operator assigns the current array element to $variable. An example will make the loop structure more clear–

Example 14.8: foreach loop in PHP

```
<?php
    $a = array("Item1", "Item2", "Item3");
    foreach($a as $v) {
      echo "Current array element is ". $v . "<br>";
    }
?>
```

In this example, the array $a is iterated by the foreach loop and each array element is assigned to the variable $v using 'as' keyword.



CHECK YOUR PROGRESS

Q.5: What will be the output of the following PHP code?

```
<?php
$fruits = array ("apple", "orange", "banana");
    echo (next($fruits));
    echo (next($fruits));
?>
```

a) orangebanana b) appleorange

c) orangeorange d) appleappl

Q.6: What will be the output of the following PHP code?

```
<?php
$i = 0
while ($i < 3)
{
    $i++;
}
print $i;
?>
```

a) 2 b) 3 c) 0 d) 1

Q.7: What will be output of the following PHP code :

```
<?php
$i = 0;
for ($i)
{
    print $i;
}
?>
```

a) 0 b) Infinite loop

c) No output d) error

Q.8: What will be output of the following PHP code?

```
<?php
 $numbers = array(9, 2, 1);
 foreach ($numbers as $value){
    echo "$value ";
}
?>
```

a) 9 2 1 b) 9 2 c) 2 d) 1



EXERCISE 14.7

Create three arrays $student, $parent and $address. The first array $student contains names of 5 students, $parent contains the name of guardian of each respective student and $address contains the address of the student. Using foreach loop display the record of each student along with his/her parent name and address.



14.8 LET US SUM UP

- Branching or conditionals or conditional statements provide the program the capability to make some logical decision on its own and perform

some specific tasks (execute a block of code) depending on whether a certain condition is satisfied or not

- If-Else statement is used to make binary decision. i.e. if-else breaks or branches the program into two different execution paths.
- Switch is a multi-option branching mechanism where a selection is made over a collection of cases for a particular condition and appropriate code block is executed if a case is satisfied, otherwise default code block gets executed.
- The looping in PHP programming is a series of statements which are repeatedly executed or iterated for a certain number of times unless some condition is met.
- There are various types of loop statements available in PHP such as for loop, while loop, foreach loop and do-while.



14.9 FURTHER READING

1) Julie, C. M., Sams. *Teach Yourself PHP in 24 Hours, Techmedia*.
2) Welling, Luke; Thomson, Laura. *PHP and MySQL Web Development*. Addison-Wesley Professional.
3) http://php.net, Official website of PHP.



14.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: (a) **Ans. to Q. No. 2:** (b)
Ans. to Q. No. 3: (c) **Ans. to Q. No. 4:** (c)
Ans. to Q. No. 5: (a) **Ans. to Q. No. 6:** (b).
Ans. to Q. No. 7: (d) **Ans. to Q. No. 8:** (a)



14.11 MODEL QUESTIONS

Q.1: What is Conditional Control? Write down the syntax of various conditional control present in PHP.

Q.2: What are common usage scenarios for switch statement? Which part of switch gets executed if none of the conditions are true?

Q.3: What is a loop? Why is loop required in programming. Give proper example.

Q.4: What are the different types of loop present in PHP? Write short note on each of these.

Q.5: Write down the structure of foreach loop. Why is foreach loop is required?

Q.6: Write down the PHP code to iterate over an array of strings $arr using foreach loop. Can we achieve the same result with for loop? Write the proper code.

Q.7: What is the use case of do-while loop? Write down the differences between while and do-while loop.

Q.8: What is infinite loop? How can you write an infinite loop using **for**?

Q.9: Give a proper programming example of an infinite while loop.

Q.10: What is break and continue? Write down the differences between the two.

Q.11: Why var_dump() is used in PHP? Write down the structure of output obtained from var_dump().

Q.12: Suppose an array $a is present somewhere in the code. You need to know the size of this array. How can you do that?

*** ***** ***

UNIT 15: FUNCTIONS IN PHP

UNIT STRUCTURE

15.1 Learning Objectives

15.2 Introduction

15.3 Functions in PHP

- 15.3.1 User Defined Functions
- 15.3.2 Built in Functions
  - 15.3.2.1 Functions Associated with Variables
  - 15.3.2.2 Array Functions
  - 15.3.2.3 Date and Time Functions
  - 15.3.2.4 Math Functions
  - 15.3.2.5 String Functions

15.4 PHP Server Variables

15.5 Working with Forms in PHP

- 15.5.1 Difference between GET and POST Method

15.6 Working with Files in PHP

15.7 Let Us Sum Up

15.8 Further Reading

15.9 Answers to Check Your Progress

15.10 Model Questions

15.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define functions in PHP
- differentiate between user-defined and built-in functions in PHP
- learn about functions, return types, arguments
- learn about server variables in PHP
- interact with the user using HTML form
- handle files in PHP.

15.2 INTRODUCTION

Equipped with the knowledge of PHP fundamentals, we are now ready to dig into some intermediate PHP concepts. In this unit, we will study about PHP functions and how to utilize them to achieve code reusability. We will have a brief overview of some popular PHP libraries like Date-Time library, String library, Mathematics library etc.

Though functions provide a great way to reuse your code, objects provide better ways to achieve it more efficiently. Being a modern language, PHP provides full object oriented support just like C++ or Java. You can study OOP in PHP after you become fluent in procedural PHP.

15.3 FUNCTION IN PHP

A function is a block of code which is defined once and can be used throughout the entire program. It is the fundamental building block of procedural programming and comes very handy when we need some predefined tasks to be performed repeatedly. For example, if in a program we have to calculate average every now and then, then it is better to define it in a function rather than writing the logic every time.

Functions are broadly classified in two types– **Built-in function** and **User defined function**. Built-in functions are available with the language itself with the help of some header files, libraries etc. User defined functions are defined by the programmers and is to be shipped manually from software to software.

In the first half of this section we will discuss about user defined functions, how we can define and use them in PHP. In the latter half, we will see some of PHP's most used built-in functions.

15.3.1 User Defined function

In PHP, programmers can define their own functions using the keyword "function" followed by the function name, followed by the list of arguments(if any) within parenthesis and finally the required block of code(also called function body) within curly braces. The general syntax is as follows–

SYNTAX for Defining a Function:

```
function function_name ($arg1, $arg2,….) {
//function body}
```

Please note that, all though PHP is a case-sensitive language, the user-defined function names are not case-sensitive. Also, the identifier rule for any C-like language is applicable in PHP too. So, a function name can contain letters, numbers and underscores only. Also, a function name cannot begin with number.

Let us take a simple example of a function with name as "DisplayMessage"to print "Hello world".

```
<?php
     /* Defining a PHP Function */
     function DisplayMessage() {
        echo "Hello world";
    }
     /* Calling a PHP Function */
    DisplayMessage();
?>
```

Now, let's try to implement a function which finds the average of some numbers using a function named 'avg_num' –

```
<?php
function avg_num($numbers) {
    $sum = 0;
    foreach($numbers as $num)
    $sum += $num;
    $avg = $sum / count($numbers);
    return $avg;
}

$a = [1, 2, 3];
echo avg_num($a);
?>
```

In the above example, function *avg_num()* accepts an array of number as argument. We have iterated and counted the sum of the numbers in the array using a foreach loop (however, there's library function for that, we'll see it later) and find the average by dividing the $sum with total number of elements in the array. The result is returned with the keyword '*return*'. The above function will return result as 2.It is not mandatory to have a return type in every function. Unlike C/C++, PHP doesn't define the data type of the return value of a function.



EXERCISE 15.1

Define a function named "hello_message" which takes two string arguments. The first argument is a person's name and second one is an integer values which values are restricted to 1, 2 and 3. If 1 is passed function shows "Good Morning" followed by person's name. Similarly 2 and 3 returns "Good Afternoon" and "Good Evening" respectively followed by the person's name.

15.3.2 Built in Function

As mentioned earlier, built-in functions comes by default with the language. The programmers can simply call them just like regular functions and use them in their code. PHP is a very feature-rich language and it comes with a great collection of built-in functions. Actually, one of the main reasons of the immense popularity of PHP is due to its enormous set of libraries which are tested and modified rigorously for almost a decade. Let's see some of the popular PHP functions.

15.3.2.1 Functions Associated with Variables

Here, $a is a PHP variable.

Function	Return Type	Example	Description
var_dump()	void	var_dump($a)	Displays the data type of a variable
gettype()	string	gettype($a)	Returns the type of the variable like 'integer', 'string', 'double' etc.
Settype()	boolean	settype($a, "string)	Set type of a variable to a different type as specified by the 2nd argument
is_int, is_float, is_bool, is_string, is_double etc.	Boolean	is_int($a); it will check whether $a is ineger of not.	Checks whether a variable is of specific type
is_numeric()	boolean	is_numeric($a)	Checks whether a variable is numeric
isset()	boolean	isset($a)	Checks if a variable is set and not NULL
print_r()	mixed	print_r($a)	Shows human readable information about a variable, similar to var_dump()

Following is an example of the use of *is_int*(), *is_float(), is_numeric(), var_dump(), isset()* and *print_r()* function.

```
<?php
$a = 16;
$b = "abc";
$c=11.54;
if (is_int($a))
    echo "$a is Integer \n" ;
else
    echo "$a is not an Integer \n" ;
```

```
if (is_float($a))
    echo "$a is float \n" ;
else
    echo "$a is not float \n" ;
if (is_int($b))
    echo "$b is Integer \n" ;
else
    echo "$b is not Integer\n" ;
if (is_numeric($c))
    echo "$c is numeric\n" ;
else
    echo "$c is not numeric\n" ;
if (isset($a))
    echo "variable 'a' is set\n" ;
else
    echo "$a is not set\n" ;
var_dump($a);
echo''\n";
var_dump($b);
echo''\n";
print_r($a);
echo "$b is not Integer\n" ;
?>
```

The output of the above code is:

16 is Integer
16 is not float
abc is not Integer
11.54 is numeric
variable 'a' is set
int(16)
string(3) "abc"
16

15.3.2.2 Array Functions

Suppose, we have an array $a

Function	Return Type	Example	Description
count()	int	count($a)	Returns the total number of elements in the array
is_array()	boolean	is_array($a)	Returns true if the variable is an array
array_merge()	array	array_merge($a, $b, $c)	Combines two or more arrays into a new array.
array_push()	int	array_push($a, “item1”, “item2”)	Insert elements to the end of an array
array_pop()	mixed	array_pop($a)	Pops and returns the last element of an array
array_sum()	number	array_sum($a)	Returns the summation of elements of an array
in_array()	boolean	in_array(“String”, $a)	Finds whether an item is present in the list. Takes another parameter for STRICT mode.
sort()	boolean	sort($a)	Sorts an array in ascending order
rsort()	boolean	rsort($a)	Sorts an array in descending order
implode()	string	implode(“,” , $a)	Combines array element to form a string separated by the string specified in first parameter.

Following example shows the use of sort() and count() functions to sort the array in ascending order:

```
<?php
$a = array(4, 6, 2, 22, 11);
sort($a);
$alength = count($a);
for($x = 0; $x < $alength; $x++) {
    echo $a[$x];
```

```
    echo "<br>";
}
?>
```

The output of the above code is: 2, 4, 6, 11, 22,

Following example shows the use of *array_merge()* and *implode()* functions to merge the arrays $a1 and $a2 into $a3.

```
<?php
$a1=array("red","green");
$a2=array("blue","yellow");
$a3=array_merge($a1,$a2);
$a3length = count($a3);
for($x = 0; $x < $a3length; $x++) {
    echo $a3[$x];
    echo "<br>";
}
    echo implode(",",$a3);
?>
```

The output of the above code is:

red
green
blue
yellow
red,green,blue,yellow

15.3.2.3 Date and Time Functions

Function	Return Type	Example	Description
date()	string	date('d-m-Y')	A date function accepts two parameters. First one is the "format" which specifies the format of the output date string and second one is the timestamp. Some widely used for string characters are–

			• d – The day of the month (from 01 to 31) • D – A textual representation of a day (three letters) • j – The day of the month without leading zeros (1 to 31) • m – A numeric representation of a month (from 01 to 12) • M – A short textual representation of a month (three letters) • Y – A four digit representation of a year • y – A two digit representation of a year • a – Lowercase am or pm • A – Uppercase AM or PM • g – 12-hour format of an hour (1 to 12) • G – 24-hour format of an hour (0 to 23) • h – 12-hour format of an hour (01 to 12) • H – 24-hour format of an hour (00 to 23) • i – Minutes with leading zeros (00 to 59) • s – Seconds, with leading zeros (00 to 59)
time()	string	time()	Returns the current time measured in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)
strtotime ()	Int	strtotime(“now”); strtotime(“5 May 2019) strtotime(“+1 day”) etc.	Converts an English date string to UNIX timestamp.

Following example shows the use of *date()* and *time()* functions:

```
<?php
echo "Today's date is :";
$today = date("d/m/Y");
echo $today;
echo "<br>Today's date is :";
$today = date("D/M/y");
echo $today;
$timestamp = time();
echo "<br>",($timestamp);
echo "<br>";
echo(date("F d, Y h:i:s A", $timestamp));
?>
```

The output of the above code is:

Today's date is :07/03/2019
Today's date is :Thu/Mar/19
1551946232
March 07, 2019 08:10:32 AM

15.3.2.4 Math Functions

Function	Return Type	Example	Description
abs()	number	abs(-2), returns 2	Returns absolute values of a number
ceil()	number	ceil(0.45), returns 1	Rounds up the number to nearest integer
floor()	number	floor(0.45), returns 0	Rounds down the number to nearest integer
pi()	number	pi()	Returns the value of PI
pow()	number	pow(5, 2), returns 25	Requires two parameters– base and power. Returns the value of base raised to the power.

Rand()	Int	rand(); rand(1,10)	Returns a random number. Range can be specified using two optional parameters.
sqrt()	float	sqrt(25), returns 5	Returns square root of a number
bindec()	number	bindec(100), returns 4	Converts binary number to decimal
decbin()	number	bindec(4), returns 100	Converts decimal number to binary

Following example shows the use of *abs()*, *floor(),* pi(), *ceil(), pow(), sqrt()* functions:

```
<?php
$number1 = -8.4;
$number2 = 4;
echo(abs($number1)); echo "<br>";
echo(ceil($number1));echo "<br>";
echo(floor($number1));echo "<br>";
# PHP function to convert degree to radian value.
echo(pi());
echo "<br>";
echo(pow($number2,4) . "<br>");
echo(sqrt($number2) . "<br>");
?>
```

The output of the above code is:

8.4

–8

–9

3.1415926535898

256

2



EXERCISE 15.2

Write a PHP function “circle_area” which takes the radius of a circle as an argument and returns its area.

15.3.2.5 STRING Functions

Here, $a is a string.

Function	Return Type	Example	Description
explode()	array	explode(“ ”, “Hello World”), returns [“Hello”, “World”]	Returns an array by breaking the strings using the given delimiter.
strrev()	string	strrev($a)	Reverses a string
strtoupper()	string	strtoupper($a)	Converts a string to uppercase
strtolower()	string	strtolower($a)	Converts a string to lowercase
ucfirst()	string	ucfirst($a)	Makes first character of a string uppercase
lcfirst()	string	lcfirst($a)	Makes first character of a string lowercase
ucwords()	string	ucwords($a)	Converts the first character of each word in the string to uppercase
strlen() trim()	int string	strlen($a) trim($a)	Returns the length of string Removes whitespace from the beginning and end of a string
strops()	int	strops(“item1”, $a)	Find the first occurrence of a substring in a string
str_replace()	string	str_replace (“search”, “replace”, $a)	Replace all occurrences of a substring, with another substring in a given string

Following example shows the use of different string functions:

```
<?php
 $str="  I am     simple boy!";
echo(strtoupper($str));

echo"<br>";
print_r(explode(" ",$str));

echo"<br>";
echo(strtolower($str));

echo"<br>";
echo(strrev($str));

echo"<br>";
echo(ucfirst($str));

echo"<br>";
echo(lcfirst($str));

echo"<br>";
echo(ucwords($str));

echo"<br>";
echo(strlen($str));

echo"<br>";
echo(trim($str));

echo"<br>";
echo strpos("I love php, I love php too!","php");

echo"<br>";
echo str_replace("php","C programming","I love php,
I love php too!");
?>
```

The output of the above code is:

I AM SIMPLE BOY!

Array ([0] => [1] => [2] => I [3] => am [4] => [5] => [6] => [7] => [8] => simple [9] => boy!)

i am simple boy!

!yob elpmis ma I

I am simple boy!

I am simple boy!

I Am Simple Boy!

22

I am simple boy!

7

I love C programming, I love C programming too!



EXERCISE 15.3

Modify the "DisplayMessge" function as shown in earlier example in such a way that the "Hello" should be displayed in all capital letters, whereas rest of the sentence should be in all lowercase. Also create another function named "name_check" which checks whether a name is palindrome and call it inside "hello_message". The "name_check" function should return a Boolean value.



CHECK YOUR PROGRESS

Q.1: Unix Timestamp is calculated from:

a) 1970 b) 1990

c) 2000 d) None

Q.2: Function to convert first character of word of a string to uppercase is:

a) ucfirst() b) ucwords()

c) strtoupper() d) lcfirst()

Q.3: Which function is used to compare strings as strings in PHP?

a) startswith() b) strcmp()

c) compare() d) compareTo()

Q.4: What will be the output of the following code? If say date is 22/06/2019.

```
<?php
    echo "Today is ".date("F d, Y");
?>
```

a) Today is 22 June, 2019 b) Today is 22-06-2019
c) Today is 06-22-2019 d) Today is June 22, 2019

Q.5: What will be the output of the following PHP code:

```
<?php
    function calc($price, $tax=0)
    {
        $total = $price + ($price * $tax);
    echo $total;
    }
    calc(42);
?>
```

a) 0 b) 42 c) 84 d) Error

Q.6: What will be the output of the following PHP code:

```
<?php
    $op2 = "hi";
    function foo($op1)
    {
        echo $op1;
        echo $op2;
    }
    foo("hello");
?>
```

a) hellohi b) hellohihi c) hello d) Error

15.4 PHP SERVER VARIABLES

As discussed earlier at UNIT 13, PHP provides a pre-defined set of variables which has the global scope and is managed by the interpreter itself. We call these variables as *super global* variables or simply *superglobals*.

Superglobals can be called anywhere in the program. A list of *superglobal* variables available in PHP are given below–

Variable	Description
$GLOBALS	It is an array which contains references to all the global variables defined in the program.
$_SERVER	This is an array containing information such as headers, paths, and script locations. The entries in this array are created by the web server.
$_GET	An associative array of variables passed to the current script via the HTTP GET method. $_GET can also collect data sent in the URL.
$_POST	An associative array of variables passed to the current script via the HTTP POST method. $_POST is also widely used to pass variables.
$_FILES	An associative array of items uploaded to the current script via the HTTP POST method.
$_REQUEST	An associative array consisting of the contents of $_GET, $_POST, and $_COOKIE. $_REQUEST is used to collect data after submitting an HTML form.
$_COOKIE	An associative array of variables passed to the current script via HTTP cookies.
$_SESSION	An associative array containing session variables available to the current script.
$_ENV	An associative array of variables passed to the current script via the environment method.

Although all the superglobals variables are important, the following examples show the use of some of the superglobals.

The PHP code shown below is the use of $GLOBALS server variable. PHP stores all global variables in an array called $GLOBALS [I] where, *I* holds the name of the variable. In this example, since *x* is a variable present in $GLOBALS array, it is also accessible from outside the function which outputs value as 85.

```
<?php
$a = 45;
$b = 25;
$c = 15;

function addition() {
$GLOBALS['x'] = $GLOBALS['a'] + $GLOBALS['b'] +
$GLOBALS['c'];
}
addition();
echo $x;
?>
```

The following example shows the use of some of the elements in $_SERVER.

```
<?php

#Returns the filename of the currently executing script
echo $_SERVER['PHP_SELF'];

#Returns the name of the host server
echo $_SERVER['SERVER_NAME'];

#Returns the Host header from the current request
echo $_SERVER['HTTP_HOST'];

#Returns the name and revision of the information protocol
echo $_SERVER['SERVER_PROTOCOL'];

#Returns the path of the current script
echo $_SERVER['SCRIPT_NAME'];

#Returns the URI of the current page
echo $_SERVER['SCRIPT_URI'];
?>
```

The example below shows the use of super global variable $_REQUEST to collect the value of the input field in a form with an input

field and a submit button. When the user submits the data by clicking on "Submit" button, the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If we wish to use another PHP file to process form data, we have to replace that with the filename of our choice.

```
<html>
<body>
<form method="post" action="<?php echo
$_SERVER['PHP_SELF'];?>">
Name: <input type="text" name="name">
<input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // collect value of input field
   $fname = $_REQUEST['name'];
   if (empty($fname)) {
      echo "Name filed is empty";
   } else {
   echo $fname;
   }
}
?>
```

We can also use $_POST to collect the value of the input files instead of $_REQUEST.

Following example shows the use of $_GET to collect the data sent in URL from the html page containing a hyperlink as shown below.

```
<html>
<body>
<a href="test.php?chapter=PHP&web=lession.com">
Test_$GET</a>
</body>
</html>
```

When a user clicks on the link “Test_$GET”, the parameters “chapter” and “web” are sent to “test.php”, and we can then access their values in “test.php” with $_GET.

The example below shows the code in “test.php”:

```
<html>
<body>
<?php
echo “Learn the course “ . $_GET[‘lession’] . “ at “
. $_GET[‘web’];
?>
</body>
</html>
```

15.5 WORKING WITH FORMS IN PHP

Forms are the primary way in a website/web-application, where the user can input their data and get processed result from the server on triggering some events. i.e., in a nutshell, forms are used for getting input from the user to be submited it to the web server for processing. HTML forms take specific type of input from the user and send them to the server using HTTP POST or HTTP GET method. The data obtained via POST/GET is then processed and returned by the server. Forms in HTML are defined with the container element <form>. Each <form> element contains some form controls like *input, radio, select* etc. Let’s see how we can link an HTML form to a server side PHP script. We will first create an HTML page named ***form.html***

form.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Working with PHP form</title>
</head>
<body>
   <h3>Text Input</h3>
   <form name=”testForm” method=”POST”
```

```
        action="process_form.php">
        First name:<br>
        <input type="text" name="fname" />
        <br>
        Last name:<br>
        <input type="text" name="lname" />
        <br>
        <input type="submit" value="Send Data" />
        </form>
    </body>
    </html>
```

In *form.html* file we've created a form named *testForm* which has two very essential attributes. The method attribute specifies the HTTP method to be used while sending the data to the server. In our case, we're interested in using the POST method. Second one is *action*. This attribute specifies the path of the PHP file (or any other scripting language file) which will process the input obtained from this form. Each input control is provided a unique name. The data will be accessed in the server using these names.

Now let's see the code of *process_form.php*

process_form.php

```
<?php
   $fname = $_POST['fname'];
   $lname = $_POST['lname'];
   echo "Hello'' . $fname . '' '' . $lname . " '' .
   ''How are you ?";
?>
```

Whenever a form is submitted, the data from input controls are entered into the appropriate PHP superglobal with their *name* attribute as key and content of the *value* attribute as value of the associative array. The PHP superglobals $_GET and $_POST are used to collect form-data.. So we can access them using $_POST as we've specified the method as POST. Please not that, the above example assumes that the *form.html* and *process_form.html* both resides in the same directory. If your PHP file is in another directory, give appropriate relative path of that directory.

We've shown how to access the data from an HTML input control to server. The procedure is same for other HTML form elements like select, textbox etc.

15.5.1 Difference between GET AND POST Method

Both GET and POST are treated as $_GET and $_POST. Information sent from an HTML form with the GET method is displayed in the browser's address bar, and it has a limit (about 2000 characters) on the amount of information to send i.e. it information is visible to everyone (all variable names and values are displayed in the URL). The GET method is used only for sending non-sensitive information. It should never be used for sending passwords or any other sensitive information. On the otherhand, Information sent from a form with the POST method is invisible (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send. The information sent by POST method is secured.

15.6 WORKING WITH FILES IN PHP

File handling is another major task in server side scripting. The user uploads files to the server using in HTML form. The server copies the file from client and stores it in the server and performs further processing as required. For file related tasks, we use PHP superglobal $_FILES.

Let's see an example. Suppose, we want a HTML form where user can upload their images to the server. Here's the sample code for that –

upload_file.html

```
<!DOCTYPE html>
<html>
<head>
    <title>File upload in  PHP</title>
</head>
<body>
```

```
    <h3>File Upload</h3>
   <form name="uploadForm" method="POST"
   action="process_file.php" enctype="multipart/
   form-data">
      Choose File:<br>
      <input type="file" name="filename" />
      <br>
      <input type="submit" value="Upload" />
   </form>
</body>
</html>
```

Please note that, while uploading a form which is used for form uploading, we need to use the attribute *enctype* with value *multipart/form-data*. It specifies the content-type which allows file upload at server side.

Now, let's check the server side code–

process_file.php

```
<!doctype html>
<html>
<head>
<title>File Upload Test</title>
</head>
<body>
   <?php
      $desiredPath = 'upload/';
      if(!empty($_FILES['filename'])) {
         if(!file_exists($desiredPath)) {
         mkdir($desiredPath, 0755, true);
         }
         $path = $desiredPath .
         basename($_FILES['filename']['name']);
         if(move_uploaded_file($_FILES['filename']
         ['tmp_name'], $path))
            echo "<img src='".$path."'>";
```

```
        }
    ?>
</body>
</html>
```

Let's try to understand the code–

- We want to store the uploaded file a specific directory called 'upload'. We have saved it in a variable $desiredPath
- Then we check if user has uploaded a file in the control *filename* using the negation of empty() function on $_FILES
- If the above condition satisfies, we check whether a directory name 'upload' exists. If not, we create it with mkdir() function which takes the parameters as directory name, access mode and recursive mode on or off.
- Now we get the name of the uploaded file using basename(). Please note that files are uploaded in a temporary location, from which programmers can move them to required location. Otherwise they are deleted automatically.
- We create the full path of the required file appending $desiredPath with filename and storing the result in $path.
- Finally, we move the file from temporary location to our required path using the function move_uploaded_file().
- If the file upload is successful, the output will be displayed. For simplicity, we're assuming that only image files will be uploaded in the system which will be displayed using <img> tag.



EXERCISE 15.4

Extend the above example in such a way that the PHP file only accepts image file with extension JPEG and size below 500KB. Hint – pathinfo($path, PATHINFO_EXTENSION) and $_FILES['filename']['size']



CHECK YOUR PROGRESS

Q.7: Which of the following contains a reference to every variable which is currently available within the global scope of the script?

a) $GLOBALS b) $_SERVER

c) $_COOKIE d) $_SESSION

Q.8: When you use the $_GET variable to collect data, the data is visible to:

a) none b) only you

c) everyone d) selected few

Q.9: Which two predefined variables are used to retrieve information from forms?

a) $GET & $SET b) $_GET & $_SET

c) $__GET & $__SET d) GET & SET

Q.10: Which one of the following should not be used while sending passwords or other sensitive information?

a) GET b) POST c) REQUEST d) NEXT

Q.11: The is a super global variable that returns the filename of the currently executing script.

a) $_SERVER[“PHP”] b) #_SERVER[“SELF_PHP”]

c) $_SERVER[“SELF”] d) $_SERVER[“PHP_SELF”]

Q.12: Which of the following method sends input to a script via a URL?

a) GET b) POST

c) Both GET and POST d) None of the above



15.7 LET US SUM UP

- A function in PHP is a block of code which is defined once and can be used throughout the entire program.

- Functions are broadly classified in two types – Built-in function and User defined function.
- There are various types of built in functions in PHP such as: function associated with variables, math functions, array functions, string functions, date and time function
- PHP provides a pre-defined set of variables which has the global scope and is managed by the interpreter itself which are called as *super global* variables or simply *superglobals*.
- Various superglobal variables are $GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_REQUEST, $_COOKIE, $_SESSION, $_ENV
- There are two methods to collect input data from form such as: GET and POST. The GET method is used only for sending non-sensitive information. On the otherhand, the information sent by POST method is secured.



15.8 FURTHER READING

1) Julie, C. M., Sams. *Teach Yourself PHP in 24 Hours, Techmedia*.
2) Welling, Luke; Thomson, Laura. *PHP and MySQL Web Development*. Addison-Wesley Professional.
3) http://php.net, Official website of PHP.



15.9 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: (a) **Ans. to Q. No. 2:** (b)
Ans. to Q. No. 3: (b) **Ans. to Q. No. 4:** (d)
Ans. to Q. No. 5: (b) **Ans. to Q. No. 6:** (c).
Ans. to Q. No. 7: (a) **Ans. to Q. No. 8:** (c)
Ans. to Q. No. 9: (b) **Ans. to Q. No. 10:** (a)
Ans. to Q. No. 11: (d) **Ans. to Q. No. 12:** (a)



15.10 MODEL QUESTIONS

Q.1: Define function. Why do we need functions? What is the keyword used in PHP to define a function?

Q.2: Write down the differences between user-defined and built-in functions.

Q.3: Explain how UNIX timestamp is calculated.

Q.4: What are the different superglobals present in PHP? Explain their usage.

Q.5: What the security threats we can encounter if we allow users to upload their files to our server?

Q.6: Write a PHP script that allows the user to upload only PDF files to our server.

Q.7: Write a short bio about yourself and store it in a PHP string variable and perform the following tasks in it–

- Replace your name with your friend's name.
- Change all instances of your Address to Uppercase.
- Find the total number of words in the paragraph
- Change first character of each word to Uppercase

Q.8: Write PHP code display the present date in the following formats –

a) 01-01-2019

b) 1-1-19

c) 1 January, 2019

Q.9: Design an HTML form to ask the user to enter their name, phone, address, sex, hobbies and show the details in the form of an HTML Table.

Q.10: Write a PHP function which takes a string as input and returns the length and frequency of each word in the string in an HTML table.

*** ***** ***



REFERENCES

1) Batross, I. (2009). *Web Enabled Commercial Application Development Using HTL, DHTML, Javascript, Peril CGI*. Bpb Publications.
2) Brooks, D. R. (2017). *Functions. In Programming in HTML and PHP* (pp. 129-147). Springer, Cham.
3) Crockford, Douglas. (2008). *JavaScript: The Good Parts*. Shroff Publishers & Distributors.
4) Deitel & Deitel. *Web Technology*. Pearson Education.
5) Duckett, J. (2011). *Beginning html, xhtml, css, and javascript*. John Wiley & Sons.
6) Godbole, Achyut S., Kahate, Atul. *Wed Technologies– TCP/IP to Internet Architectures*. Tata McGraw Hill.
7) *JavaScript: The Definitive Guide*. (Edition 2) (1996). Publisher: O'Reilly Media.
8) Jackson, J. C. (2007). *Web Technologies: A Computer Science Perspective*. Pearson/Prentice Hall.
9) Julie C. M., Sams. *Teach Yourself PHP in 24 Hours, Techmedia*.
10) Kamal, Raj. *Internet and Web Technologies*. Tata McGraw Hill.
11) Haverbeke, Marijn. *Eloquent JavaScript: A Modern Introduction to Programming*. (2nd Edition).
12) Holzner, Steve. (2006). *Ajax For Dummies*.
13) Riordan, Rebecca M. (2008). *Head First Ajax: A Brain-Friendly Guide*. O'Reilly Media.
14) Roy, Uttam K. (2010). *Web Technologies*. Oxford Higher Education.
15) Sharma, Kumar Chetan; Stefanov, Stoyan; Antani, Ved; Hillar, Gastón C. *JavaScrip: Object-Oriented Programming*. Publisher: Packt Publishing Limited.
16) Sharma, Pankaj. (2011). *Introduction to Web Technology*. Katson Books.
17) Sikos, L. (2014). *Web Standards: Mastering HTML5, CSS3, and XML*. Apress.
18) Stefanov, Stoyan. *Object-Oriented JavaScript*.

19) Welling, Luke, Thomson, Laura. *PHP and MySQL Web Development, Addison-Wesley Professional*.
20) Willard, W. (2002). *HTML: A Beginner's Guide*. Osborne.
21) Zakas, Nicholas C. McPeak, Jeremy & Fawcett, Joe. (2007). *Professional Ajax*. Wrox.
22) http://php.net, Official website of PHP.
23) https://www.webopedia.com,
24) https://searchenterprisedesktop.techtarget.com/,
25) https://www.tutorialspoint.com/,
26) https://www.computerhope.com, https://www.techopedia.com
27) WWW.W3schools.com.

Name of the Paper: .....
Course Code: .....

Centre for Internal Quality Assurance (CIQA)
Krishna Kanta Handiqui State Open University
City Office: Housefed Complex, Guwahati-781006

Learner's Feedback on Course

Dear Learner,

Regarding the course as mentioned above, we would like to know your opinions and comments so as to improve the quality of self learning materials in future. Please respond to the following statements by ticking the number you feel most reflect your opinion. After completion of the additional comments, please detach the page and send/mail the same to us at the address given below.

The Director, Centre for Internal Quality Assurance, KKHandiqui State Open University
Housefed Complex, Dispur, Guwahati-781006
(E-mail id: ciqa@kkhsou.in)

1) Approximately how many hours did you spend for studying the units in the course? ☐☐☐

2) Please give your opinions (by ✓ mark) to the following items based on your reading of the block:

Sl. No.	Statements	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
I)	The SLMs of the course fulfil the learning objectives set out in the beginning of each unit					
II)	The units of the SLM could satisfy my academic needs and expectations					
III)	The Unit writers have excellent knowledge about the course contents					
IV)	Language and contents of the units were presented at a level which I could readily understand					
V)	Ample opportunity for participation in the activities provided in the units					
VI)	Used enough Illustrations (Diagrams, tables etc.) for conceptual clarity					
VII)	Quality of content is engaging, relevant, and up-to-date					
VIII)	The self check questions are very helpful					
IX)	The Possible/model questions and the answers to check my progress have benefited me a lot					

Additional Comments: *(Please feel free to provide your open comments)*

1) Which aspects of the SLM, according to you, worked well?

.....

.....

2) What sort of changes/improvements do you feel KKHSOU could implement to improve the overall quality of the SLM?

.....

.....

Thank you for taking the time to complete this form.