

UNIT IV

PHP and XML

An introduction to PHP: PHP- Using PHP- Variables- Program control- Built-in functions- Form Validation- Regular Expressions - File handling – Cookies - Connecting to Database. XML: Basic XML- Document Type Definition- XML Schema DOM and Presenting XML, XML Parsers and Validation, XSL and XSLT Transformation, News Feed (RSS and ATOM).

TEXT BOOK:

1.Deitel and Deitel and Nieto, —Internet and World Wide Web - How to Programll, Prentice Hall, 5th Edition, 2011.

REFERENCES:

- 1.Stephen Wynkoop and John Burke —Running a Perfect Website, QUE, 2nd Edition,1999.
- 2.Chris Bates, Web Programming – Building Intranet Applications, 3rd Edition, Wiley Publications, 2009.
- 3.Jeffrey C and Jackson, —Web Technologies A Computer Science Perspective, Pearson Education, 2011.
- 4.Gopalan N.P. and Akilandeswari J., —Web Technologyll, Prentice Hall of India, 2011.
- 5.UttamK.Roy, —Web Technologiesll, Oxford University Press, 2011.

4.1 PHP: INTRODUCTION TO PHP-USING PHP

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible –

- Simplicity
- Efficiency

- Security
- Flexibility
- Familiarity

"Hello World" Script in PHP

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <?php echo "Hello, World!";?>
  </body>
</html>
```

Output

Hello, World!

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser.

All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ATE are recognised by the PHP Parser.

```
<?php PHP code goes here ?>
<? PHP code goes here ?>
<script language = "php"> PHP code goes here </script>
```

A most common tag is the <?php...?>

4.2 VARIABLES

The main way to store information in the middle of a PHP program is by using a variable. Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign (\$).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

PHP has a total of eight data types which we use to construct our variables –

- **Integers** – are whole numbers, without a decimal point, like 4195.
- **Doubles** – are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** – have only two possible values either true or false.
- **NULL** – is a special type that only has one value: NULL.
- **Strings** – are sequences of characters, like 'PHP supports string operations.'
- **Arrays** – are named and indexed collections of other values.
- **Objects** – are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** – are special variables that hold references to resources external to PHP (such as database connections).

The first five are *simple types*, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

4.2.1 Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type. They correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so –

```
$int_var = 12345;  
$another_int = -12345 + 12345;
```

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.

For most common platforms, the largest integer is (2**31 . 1) (or 2,147,483,647), and the smallest (most negative) integer is . (2**31 . 1) (or .2,147,483,647).

4.2.2 Doubles

They look like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code –

```
<?php  
  
$many = 2.2888800;  
  
$many_2 = 2.2111200;  
  
$few = $many + $many_2;  
  
print("$many + $many_2 = $few <br>");  
  
?>
```

Output

2.28888 + 2.21112 = 4.5

4.2.3 Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so –

```
if (TRUE)  
    print("This will always print<br>");  
  
else  
    print("This will never print<br>");
```

4.2.4 Interpreting other types as Booleans

The rules for determine the "truth" of any value not already of the Boolean type –

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
- Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

```
$true_num = 3 + 0.14159;  
$true_str = "Tried and true"  
$true_array[49] = "An array element";  
$false_array = array();  
$false_null = NULL;  
$false_num = 999 - 999;  
$false_str = "";
```

4.2.5 NULL

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this –

```
$my_var = NULL;
```

The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed –

```
$my_var = null;
```

A variable that has been assigned NULL has the following properties –

- It evaluates to FALSE in a Boolean context.
- It returns FALSE when tested with IsSet() function.

4.2.6 Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = 'This is a somewhat longer, singly quoted string';
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php
$variable = "name";
$literally = 'My $variable will not print!';
print($literally);
print "<br>";
$literally = "My $variable will print!";
print($literally);
?>
```

Output

My \$variable will not print!

My name will print

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP –

- Certain character sequences beginning with backslash (\) are replaced with special characters

- Variable names (starting with \$) are replaced with string representations of their values.

The escape-sequence replacements are –

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \\$ is replaced by the dollar sign itself (\$)
- \" is replaced by a single double-quote ("")
- \\ is replaced by a single backslash (\)

4.2.7 Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types –

- Local variables
- Function parameters
- Global variables
- Static variables

4.2.8 Variable Naming

Rules for naming a variable is –

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , (,) . & , etc
- There is no size limit for variables.

4.3 PROGRAM CONTROL

4.3.1 Decision Making

PHP supports following three decision making statements –

- **if...else statement** – use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** – is used with the if...else statement to execute a set of code if **one** of the several condition is true

- **switch statement** – is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

The If...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

```
if (condition)
    code to be executed if condition is true;
else
    code to be executed if condition is false;
```

Example

The following example will output "Have a nice weekend!" if the current day is Friday, Otherwise, it will output "Have a nice day!":

```
<html>
<body>
<?php
$d = date("D");
if ($d == "Fri")
    echo "Have a nice weekend!";
else
    echo "Have a nice day!";
?>
</body>
</html>
```

Output

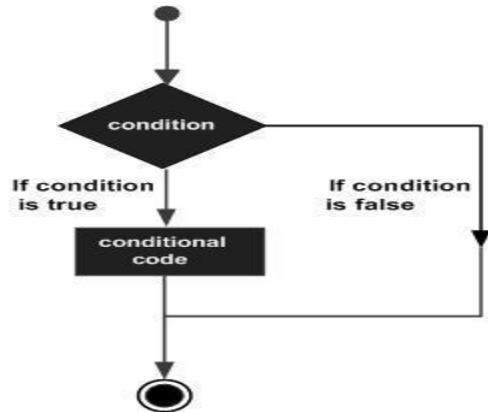


Fig: 4.1 Decision making

Have a nice weekend!

The Elseif Statement

If you want to execute some code if one of the several conditions are true use the elseif statement

Syntax

```
if (condition)
    code to be executed if condition is true;
elseif (condition)
    code to be executed if condition is true;
else
    code to be executed if condition is false;
```

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise, it will output "Have a nice day!" –

```
<html>
    <body>
        <?php
            $d = date("D");
            if ($d == "Fri")
                echo "Have a nice weekend!";
            elseif ($d == "Sun")
                echo "Have a nice Sunday!";
            else
                echo "Have a nice day!";
        ?>
    </body>
</html>
```

Output:

Have a nice Weekend!

The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax

```
switch (expression){  
    case label1:  
        code to be executed if expression = label1;  
        break;  
    case label2:  
        code to be executed if expression = label2;  
        break;  
    default:  
        code to be executed  
        if expression is different  
        from both label1 and label2;  
}
```

Example

- The **switch** statement works in an unusual way. First it evaluates given expression then seeks a label to match the resulting value.
- If a matching value is found then the code associated with the matching label will be executed or if none of the label matches then statement will execute any specified default code.

```
<html>  
    <body>  
        <?php  
            $d = date("D");  
            switch ($d){  
                case "Mon":
```

```
echo "Today is Monday";
break;

    case "Tue":
echo "Today is Tuesday";
break;

    case "Wed":
echo "Today is Wednesday";
break;

    case "Thu":
echo "Today is Thursday";
break;

    case "Fri":
echo "Today is Friday";
break;

    case "Sat":
echo "Today is Saturday";
break;

    case "Sun":
echo "Today is Sunday";
break;

    default:
echo "Wonder which day is this ?";

}

?>

</body>

</html>
```

Output:

Today is Monday

4.3.2 Looping Statements

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

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

The for loop statement

The for statement is used when you know how many times you want to execute a statement or a block of statements.

Syntax

```
for (initialization; condition; increment)  
    code to be executed;  
}
```

The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it \$i.

Example

The following example makes five iterations and changes the assigned value of two variables on each pass of the loop –

```
<html>  
<body>  
    <?php  
        $a = 0;  
        $b = 0;  
        for( $i = 0; $i<5; $i++ ) {  
            $a += 10;  
        }  
    ?>
```

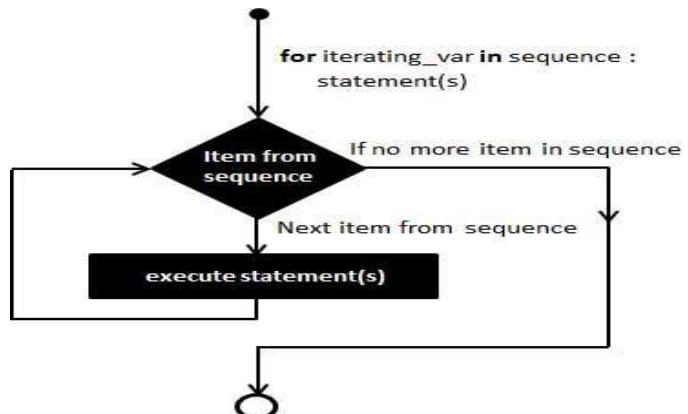


Fig: 4.2 for loop statement

```

$b += 5;
}

echo ("At the end of the loop a = $a and b = $b" );

?>

</body>

</html>

```

Output:

At the end of the loop a = 50 and b = 25

The while loop statement

- The while statement will execute a block of code if and as long as a test expression is true.
- If the test expression is true then the code block will be executed.
- After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

Syntax

```

while (condition) {

    code to be executed;

}

```

Example

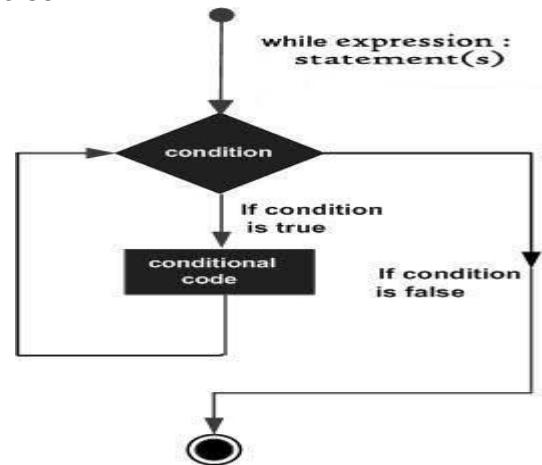
This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

Fig:4.3 while loop statement

```

<html>
<body>
<?php
$i = 0;
$num = 50;
while( $i < 10) {
    $num--;

```



```

    $i++;
}

echo ("Loop stopped at i = $i and num = $num" );
?>
</body>

</html>
```

Output

Loop stopped at i = 10 and num = 40

The do...while loop statement

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax

```

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

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10 –

```

<html>
    <body>
        <?php
            $i = 0;
            $num = 0;
            do {
                $i++;
            }
            while( $i < 10 );
            echo ("Loop stopped at i = $i" );
        </body>
    </html>
```

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

Output:

Loop stopped at i = 10

The foreach loop statement

- The foreach statement is used to loop through arrays.
- For each pass the value of the current array element is assigned to \$value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

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

Example

```
<html>  
    <body>  
        <?php  
            $array = array( 1, 2, 3, 4, 5);  
            foreach( $array as $value ) {  
                echo "Value is $value <br />";  
            }  
        ?>  
    </body>  
</html>
```

Output:

Value is 1

Value is 2

Value is 3

Value is 4

Value is 5

The break statement

- The PHP **break** keyword is used to terminate the execution of a loop prematurely.
- The **break** statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out.
- After coming out of a loop immediate statement to the loop will be executed.

Example

In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

```
<html>
  <body>
    <?php
      $i = 0;
      while( $i < 10 ) {
        $i++;
        if( $i == 3 )break;
      } echo ("Loop stopped at i = $i" );
    ?>
  </body>
</html>
```

Output:

Loop stopped at i = 3

The continue statement

- The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.
- Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test.
- For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.

Example

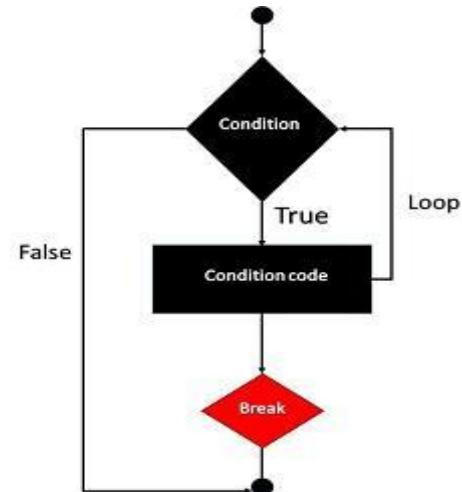


Fig: 4.4 Break statements

In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

```
<html>
<body>
<?php
$array = array( 1, 2, 3, 4, 5);
foreach( $array as $value ) {
    if( $value == 3 )continue;
    echo "Value is $value <br />";
}
?>
</body>
```

Fig: 4.5 continue statement

```
</html>
```

Output:

Value is 1

Value is 2

Value is 4

Value is 5

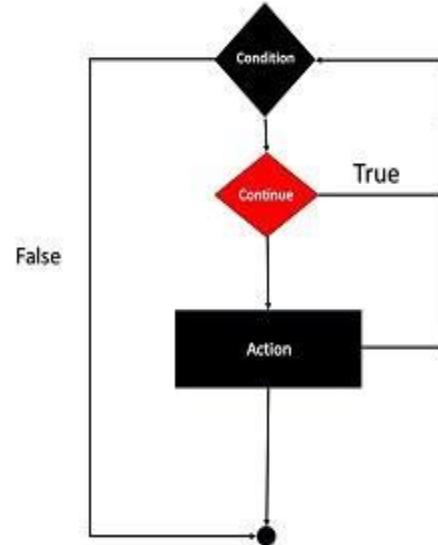
4.4 BUILT IN FUNCTIONS:

4.4.1 PHP Functions

- PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.
- In PHP, we can define **Conditional function**, **Function within Function** and **Recursive function** also.

4.4.2 Advantage of PHP Functions

- **Code Reusability:** PHP functions are defined only once and can be invoked many times, like in other programming languages.
- **Less Code:** It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.



- **Easy to understand:** PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

4.4.3 PHP User-defined Functions

Syntax

```
function functionname(){
    //code to be executed
}
```

Note: Function name must be start with letter and underscore only like other labels in PHP. It can't be start with numbers or special symbols.

PHP Functions Example

File: function1.php

```
<?php
function sayHello(){
    echo "Hello PHP Function";
}
sayHello();//calling function
?>
```

Output:

Hello PHP Function

4.4.4 PHP Function Arguments

- Information pass in PHP function through arguments which is separated by comma.
- PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

Example

File: functionarg.php

```
<?php
function sayHello($name){
```

```

echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>

```

Output:

Hello Sonoo

Hello Vimal

Hello John

Example

File: functionarg2.php

```

<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>

```

Output:

Hello Sonoo, you are 27 years old

Hello Vimal, you are 29 years old

Hello John, you are 23 years old

4.4.5 PHP Call By Reference

- Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.
- By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

Example:

File: functionref.php

```
<?php  
function adder(&$str2)  
{  
    $str2 .= 'Call By Reference';  
}  
  
$str = 'Hello ';  
adder($str);  
echo $str;  
?>
```

Output:

Hello Call By Reference

4.4.6 PHP Function: Default Argument Value

- We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument value in PHP function.

File: functiondefaultarg.php

```
<?php  
function sayHello($name="Sonoo"){  
    echo "Hello $name<br/>";  
}  
  
sayHello("Rajesh");  
  
sayHello();//passing no value  
sayHello("John");  
?>
```

Output:

Hello Rajesh

Hello Sonoo

Hello John

4.4.7 PHP Function: Returning Value

Example:

File: functiondefaultarg.php

```
<?php  
function cube($n){  
    return $n*$n*$n;  
}  
echo "Cube of 3 is: ".cube(3);  
?>
```

Output:

Cube of 3 is: 27

4.5 FORM VALIDATION

PHP Form Validation Example

* required field

Name: *

E-mail: *

Website:

Comment:


Gender: Female Male Other *

Input:

The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

4.5.1 Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

Name: <input type="text" name="name">

E-mail: <input type="text" name="email">

Website: <input type="text" name="website">

Comment: <textarea name="comment" rows="5" cols="40"></textarea>

4.5.2 Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

Gender:

<input type="radio" name="gender" value="female">Female

<input type="radio" name="gender" value="male">Male

<input type="radio" name="gender" value="other">Other

4.5.3 The Form Element

The HTML code of the form looks like this:

<form method="post" action="<?php echo htmlspecialchars(\$_SERVER["PHP_SELF"]);?>">

When the form is submitted, the form data is sent with method="post".

4.5.4 What is the \$_SERVER["PHP_SELF"] variable?

- The \$_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.
- So, the \$_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

4.5.5 What is the htmlspecialchars() function?

- The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

4.5.6 Big Note on PHP Form Security

- The \$_SERVER["PHP_SELF"] variable can be used by hackers!
- If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.
- Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.
- Assume we have the following form in a page named "test_form.php":

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

- Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<form method="post" action="test_form.php">
```

- However, consider that a user enters the following URL in the address bar:

http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E

- In this case, the above code will be translated to:

```
<form method="post" action="test_form.php/"><script>alert('hacked')</script>
```

- This code adds a script tag and an alert command. And when the page loads, the JavaScript code will be executed (the user will see an alert box). This is just a simple and harmless example how the PHP_SELF variable can be exploited.
- Be aware of that any JavaScript code can be added inside the <script> tag! A hacker can redirect the user to a file on another server, and that file can hold malicious code that can alter the global variables or submit the form to another address to save the user data, for example.

How To Avoid \$_SERVER["PHP_SELF"] Exploits?

\$_SERVER["PHP_SELF"] exploits can be avoided by using the htmlspecialchars() function.

The form code should look like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

- The htmlspecialchars() function converts special characters to HTML entities. Now if the user tries to exploit the PHP_SELF variable, it will result in the following output:

```
<form method="post"
action="test_form.php">&lt;script&ampgtalert('hacked')&lt;/script&ampgt">
```

Validate Form Data With PHP

- The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.
- When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

```
<script>location.href('http://www.hacked.com')</script>
```

- this would not be executed, because it would be saved as HTML escaped code, like this:
<script&gtlocation.href('http://www.hacked.com')</script&gt;

- The code is now safe to be displayed on a page or inside an e-mail.
- Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function).
- Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

Example

```
<?php

// define variables and set to empty values

$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = test_input($_POST["name"]);
    $email = test_input($_POST["email"]);
    $website = test_input($_POST["website"]);
    $comment = test_input($_POST["comment"]);
    $gender = test_input($_POST["gender"]);
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
}
```

```

return $data;
}
?>
```

4.6 REGULAR EXPRESSIONS:

- Regular expressions are nothing more than a sequence or pattern of characters itself. They provide the foundation for pattern-matching functionality.
- Using regular expression you can search a particular string inside another string, you can replace one string by another string and you can split a string into many chunks.
- PHP offers functions specific to two sets of regular expression functions, each corresponding to a certain type of regular expression. You can use any of them based on your comfort.
 - POSIX Regular Expressions
 - PERL Style Regular Expressions

4.6.1 POSIX Regular Expressions

- The structure of a POSIX regular expression is not dissimilar to that of a typical arithmetic expression: various elements (operators) are combined to form more complex expressions.
- The simplest regular expression is one that matches a single character, such as g, inside strings such as g, haggle, or bag.

Brackets

Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Sr.No	Expression & Description
1	[0-9]- It matches any decimal digit from 0 through 9.
2	[a-z]- It matches any character from lower-case a through lowercase z
3	[A-Z]- It matches any character from uppercase A through uppercase Z.
4	[a-Z]- It matches any character from lowercase a through uppercase Z.

- The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from b through v.

Quantifiers

- The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character having a specific connotation. The +, *, ?, {int. range}, and \$ flags all follow a character sequence.

Sr.No	Expression & Description
1	p+
2	p*
3	p?
4	p{N}
5	p{2,3}
6	p{2, }
7	p\$
8	^p

Examples

Following examples will clear your concepts about matching characters.

Sr.No	Expression & Description
1	[^a-zA-Z]
2	p.p
3	^.{2}\$
4	(.*)
5	p(hp)*

Predefined Character Ranges

- For your programming convenience several predefined character ranges, also known as character classes, are available. Character classes specify an entire range of characters, for example, the alphabet or an integer set –

Sr.No	Expression & Description	
1	<code>[:alpha:]</code>	It matches any string containing alphabetic characters aA through zZ.
2	<code>[:digit:]</code>	It matches any string containing numerical digits 0 through 9.
3	<code>[:alnum:]</code>	It matches any string containing alphanumeric characters aA through zZ and 0 through 9.
4	<code>[:space:]</code>	It matches any string containing a space.

PHP's Regexp POSIX Functions

PHP currently offers seven functions for searching strings using POSIX-style regular expressions –

Sr.No	Functions & Description	
1	<code>ereg()</code>	The <code>ereg()</code> function searches a string specified by <code>string</code> for a string specified by <code>pattern</code> , returning true if the pattern is found, and false otherwise.
2	<code>ereg_replace()</code>	The <code>ereg_replace()</code> function searches for <code>string</code> specified by <code>pattern</code> and replaces <code>pattern</code> with <code>replacement</code> if found.
3	<code>eregi()</code>	The <code>eregi()</code> function searches throughout a string specified by <code>pattern</code> for a string specified by <code>string</code> . The search is not case sensitive.
4	<code>eregi_replace()</code>	The <code>eregi_replace()</code> function operates exactly like <code>ereg_replace()</code> , except that the search for <code>pattern</code> in <code>string</code> is not case sensitive.
5	<code>split()</code>	The <code>split()</code> function will divide a string into various elements, the boundaries of each element based on the occurrence of <code>pattern</code> in <code>string</code> .
6	<code>spliti()</code>	The <code>spliti()</code> function operates exactly in the same manner as its sibling <code>split()</code> , except that it is not case sensitive.
7	<code>sql_regcase()</code>	The <code>sql_regcase()</code> function can be thought of as a utility function, converting each character in the input parameter

		string into a bracketed expression containing two characters.
--	--	---

PERL Style Regular Expressions

- Perl-style regular expressions are similar to their POSIX counterparts. The POSIX syntax can be used almost interchangeably with the Perl-style regular expression functions. In fact, you can use any of the quantifiers introduced in the previous POSIX section.

Sr.No	Functions & Description	
1	preg_match()	The preg_match() function searches string for pattern, returning true if pattern exists, and false otherwise.
2	preg_match_all()	The preg_match_all() function matches all occurrences of pattern in string.
3	preg_replace()	The preg_replace() function operates just like ereg_replace(), except that regular expressions can be used in the pattern and replacement input parameters.
4	preg_split()	The preg_split() function operates exactly like split(), except that regular expressions are accepted as input parameters for pattern.
5	preg_grep()	The preg_grep() function searches all elements of input_array, returning all elements matching the regexp pattern.
6	preg_quote()	Quote regular expression characters

Meta characters

- A meta character is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning.
- For instance, you can search for large money sums using the '\d' meta character: **/([\d]+)000/**, Here \d will search for any string of numerical character.
- Following is the list of meta characters which can be used in PERL Style Regular Expressions.

Modifiers

Several modifiers are available that can make your work with regexps much easier, like case sensitivity, searching in multiple lines etc.

Modifier Description

- i Makes the match case insensitive
- m Specifies that if the string has newline or carriage return characters, the ^ and \$ operators will now match against a newline boundary, instead of a string boundary
- o Evaluates the expression only once
- s Allows use of . to match a newline character
- x Allows you to use white space in the expression for clarity
- g Globally finds all matches
- cg Allows a search to continue even after a global match fails

PHP's Regexp PERL Compatible Functions

PHP offers following functions for searching strings using Perl-compatible regular expressions –

4.7 FILE HANDLING

Functions related to files –

- Opening a file
- Reading a file
- Writing a file
- Closing a file

4.7.1 Opening and Closing Files

- The PHP fopen() function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate.
- File modes can be specified as one of the six options in this table.

Sr.No		Mode & Purpose
1	r	<ul style="list-style-type: none">● Opens the file for reading only.● Places the file pointer at the beginning of the file.
2	r+	<ul style="list-style-type: none">● Opens the file for reading and writing.● Places the file pointer at the beginning of the file.
3	w	<ul style="list-style-type: none">● Opens the file for writing only.● Places the file pointer at the beginning of the file. And truncates the file to zero length.● If file does not exist then it attempts to create a file.
4	w+	<ul style="list-style-type: none">● Opens the file for reading and writing only.● Places the file pointer at the beginning of the file. And truncates the file to zero length.● If file does not exist then it attempts to create a file.

5	a	<ul style="list-style-type: none"> • Opens the file for writing only. • Places the file pointer at the end of the file. • If file does not exist then it attempts to create a file.
---	---	--

- If an attempt to open a file fails then fopen returns a value of false otherwise it returns a file pointer which is used for further reading or writing to that file.
- After making changes to the opened file it is important to close it with the fclose() function. The fclose() function requires a file pointer as its argument and then returns true when the closure succeeds or false if it fails.

4.7.2 Reading a file

- Once a file is opened using fopen() function it can be read with a function called fread(). This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.
- The file's length can be found using the filesize() function which takes the file name as its argument and returns the size of the file expressed in bytes.
- So here are the steps required to read a file with PHP.
 - Open a file using fopen() function.
 - Get the file's length using filesize() function.
 - Read the file's content using fread() function.
 - Close the file with fclose() function.
- The following example assigns the content of a text file to a variable then displays those contents on the web page.

```

<html>
  <head>
    <title>Reading a file using PHP</title>
  </head>
  <body>
    <?php
      $filename = "tmp.txt";
      $file = fopen( $filename, "r" );
      if( $file == false ) {
        echo ( "Error in opening file" );
        exit();
      }
      $filesize = filesize( $filename );
    </?php>
  </body>
</html>

```

```

$filetext = fread( $file, $filesize );
fclose( $file );
echo ( "File size : $filesize bytes" );
echo ( "<pre>$filetext</pre>" );
?>
</body>
</html>

```

Output:

```

File size : 278 bytes

The PHP Hypertext Preprocessor (PHP) is a programming
language that allows web developers to create dynamic
content that interacts with databases.
PHP is basically used for developing web based software
applications. This tutorial helps you to build your base
with PHP.

```

4.7.3 Writing a file

- A new file can be written or text can be appended to an existing file using the PHP fwrite() function.
- This function requires two arguments specifying a file pointer and the string of data that is to be written.
- Optionally a third integer argument can be included to specify the length of the data to write. If the third argument is included, writing would stop after the specified length has been reached.
- The following example creates a new text file then writes a short text heading inside it. After closing this file its existence is confirmed using file_exist() function which takes file name as an argument

```

<?php
$filename = "/home/user/guest/newfile.txt";
$file = fopen( $filename, "w" );
if( $file == false ) {
echo ( "Error in opening new file" );
exit();
}
fwrite( $file, "This is a simple test\n" );

```

```

fclose( $file );

?>

<html>
    <head>
        <title>Writing a file using PHP</title>
    </head>
    <body>
        <?php
            $filename = "newfile.txt";
            $file = fopen( $filename, "r" );
            if( $file == false ) {
                echo ( "Error in opening file" );
                exit();
            }
            $filesize = filesize( $filename );
            $filetext = fread( $file, $filesize );
            fclose( $file );
            echo ( "File size : $filesize bytes" );
            echo ( "$filetext" );
            echo("file name: $filename");
        ?>
    </body>
</html>

```

Output:

```

File size : 23 bytes
This is a simple test
file name: newfile.txt

```

4.8 COOKIES

- Cookies are text files stored on the client computer and they are kept for use tracking purpose. PHP transparently supports HTTP cookies.
- There are three steps involved in identifying returning users –
 - Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
 - Browser stores this information on local machine for future use.
 - When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

4.8.1 The Anatomy of a Cookie

- Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser). A PHP script that sets a cookie might send headers that look something like this –

HTTP/1.1 200 OK

Date: Fri, 04 Feb 2000 21:03:38 GMT

Server: Apache/1.3.9 (UNIX) PHP/4.0b3

Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;

path=/; domain=tutorialspoint.com

Connection: close

Content-Type: text/html

- The Set-Cookie header contains a name value pair, a GMT date, a path and a domain. The name and value will be URL encoded.
- The expires field is an instruction to the browser to "forget" the cookie after the given time and date.
- If the browser is configured to store cookies, it will then keep this information until the expiry date.
- If the user points the browser at any page that matches the path and domain of the cookie, it will resend the cookie to the server. The browser's headers might look something like this –

GET / HTTP/1.0

Connection: Keep-Alive

User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)

Host: zink.demon.co.uk:1126

*Accept: image/gif, */**

Accept-Encoding: gzip

Accept-Language: en

Accept-Charset: iso-8859-1,;utf-8*

Cookie: name=xyz

- A PHP script will then have access to the cookie in the environmental variables `$_COOKIE` or `$HTTP_COOKIE_VARS[]` which holds all cookie names and values. Above cookie can be accessed using `$HTTP_COOKIE_VARS["name"]`.

4.8.2 Setting Cookies with PHP

- PHP provided `setcookie()` function to set a cookie. This function requires upto six arguments and should be called before `<html>` tag. For each cookie this function has to be called separately.

```
setcookie(name, value, expire, path, domain, security);
```

- Here is the detail of all the arguments –
 - **Name** – This sets the name of the cookie and is stored in an environment variable called `HTTP_COOKIE_VARS`. This variable is used while accessing cookies.
 - **Value** – This sets the value of the named variable and is the content that you actually want to store.
 - **Expiry** – This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
 - **Path** – This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
 - **Domain** – This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
 - **Security** – This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.
- Following example will create two cookies name and age these cookies will be expired after one hour.

```
<?php  
  
setcookie("name", "John Watkin", time()+3600, "/", "", 0);  
  
setcookie("age", "36", time()+3600, "/", "", 0);  
  
?>  
  
<html>  
  
<head>
```

```

<title>Setting Cookies with PHP</title>
</head>
<body>
<?php echo "Set Cookies"?>
</body>
</html>

```

4.8.3 Accessing Cookies with PHP

- PHP provides many ways to access cookies. Simplest way is to use either `$_COOKIE` or `$HTTP_COOKIE_VARS` variables. Following example will access all the cookies set in above example.

```

<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
echo $_COOKIE["name"]. "<br />";
/* is equivalent to */
echo $HTTP_COOKIE_VARS["name"]. "<br />";
echo $_COOKIE["age"] . "<br />";
/* is equivalent to */
echo $HTTP_COOKIE_VARS["age"] . "<br />";
?>
</body>
</html>

```

- You can use `isSet()` function to check if a cookie is set or not.

```

<html>
<head>
<title>Accessing Cookies with PHP</title>

```

```

</head>
<body>
<?php
if( isset($_COOKIE["name"]))
    echo "Welcome " . $_COOKIE["name"] . "<br />";
else
    echo "Sorry... Not recognized" . "<br />";
?> </body>
</html>

```

4.8.4 Deleting Cookie with PHP

- Officially, to delete a cookie you should call setcookie() with the name argument only but this does not always work well, however, and should not be relied on.
- It is safest to set the cookie with a date that has already expired –

```

<?php
    setcookie( "name", "", time()- 60, "/", "", 0);
    setcookie( "age", "", time()- 60, "/", "", 0);
?>
<html>
<head>
    <title>Deleting Cookies with PHP</title>
</head>
<body>
    <?php echo "Deleted Cookies" ?>
</body>
</html>

```

4.9 CONNECTION TO DATABASE

Since PHP 5.5, **mysql_connect()** extension is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_connect()**

- o **PDO::__construct()**

4.9.1 PHP mysqli_connect()

PHP mysqli_connect() function is used to connect with MySQL database. It returns resource if connection is established or null.

Syntax

```
resource mysqli_connect (server, username, password)
```

4.9.2 PHP mysqli_close()

PHP mysqli_close() function is used to disconnect with MySQL database. It returns true if connection is closed or false.

Syntax

```
bool mysqli_close(resource $resource_link)
```

4.9.3 PHP MySQL Connect Example

Example

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

Output:

Connected successfully

4.10 XML: BASIC XML

XML stands for **Extensible Markup Language** and is a text-based markup language derived from Standard Generalized Markup Language (SGML).

4.11 DOCUMENT TYPE DEFINITIONS

- The XML Document Type Declaration, 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.

Syntax

Basic syntax of a DTD is as follows –

```
<!DOCTYPE element DTD identifier
```

```
[
```

```
declaration1
```

```
declaration2
```

```
.....
```

```
]>
```

In the above syntax,

- The DTD starts with `<!DOCTYPE` delimiter.
- An `element` tells the parser to parse the document from the specified root element.
- `DTD identifier` is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called External Subset.
- The square brackets [] enclose an optional list of entity declarations called Internal Subset.

4.11.1 Internal DTD

- A DTD is referred to as an internal DTD if elements are declared within the XML files.
- To refer it as internal DTD, `standalone` attribute in XML declaration must be set to yes. This means, the declaration works independent of an external source.

Syntax

Following is the syntax of internal DTD –

```
<!DOCTYPE root-element [element-declarations]>
```

- where `root-element` is the name of root element and `element-declarations` is where you declare the elements.

Example

Following is a simple example of internal DTD –

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address [
    <!ELEMENT address (name,company,phone)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT company (#PCDATA)>
    <!ELEMENT phone (#PCDATA)>
]>
<address>
    <name>Tanmay Patil</name>
    <company>TutorialsPoint</company>
    <phone>(011) 123-4567</phone>
</address>
```

Let us go through the above code –

- **Start Declaration** – Begin the XML declaration with the following statement.

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

- **DTD** – Immediately after the XML header, the document type declaration follows, commonly referred to as the DOCTYPE –

```
    <!DOCTYPE address [
```

- The DOCTYPE declaration has an exclamation mark (!) at the start of the element name. The DOCTYPE informs the parser that a DTD is associated with this XML document.
- **DTD Body** – The DOCTYPE declaration is followed by body of the DTD, where you declare elements, attributes, entities, and notations.

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone_no (#PCDATA)>
```

- Several elements are declared here that make up the vocabulary of the <name> document. <!ELEMENT name (#PCDATA)> defines the element name to be of type "#PCDATA". Here #PCDATA means parse-able text data.

- **End Declaration** – Finally, the declaration section of the DTD is closed using a closing bracket and a closing angle bracket (>]). This effectively ends the definition, and thereafter, the XML document follows immediately.

Rules

- The document type declaration must appear at the start of the document (preceded only by the XML header) – it is not permitted anywhere else within the document.
- Similar to the DOCTYPE declaration, the element declarations must start with an exclamation mark.
- The Name in the document type declaration must match the element type of the root element.

4.11.2 External DTD

- In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as no. This means, declaration includes information from the external source.

Syntax

Following is the syntax for external DTD –

```
<!DOCTYPE root-element SYSTEM "file-name">
```

where file-name is the file with .dtd extension.

Example

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
    <name>Tanmay Patil</name>
    <company>TutorialsPoint</company>
    <phone>(011) 123-4567</phone>
</address>
```

- The content of the DTD file address.dtd is as shown –

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
```

```
<!ELEMENT phone (#PCDATA)>
```

4.11.3 Types

System Identifiers

- A system identifier enables you to specify the location of an external file containing DTD declarations. Syntax is as follows –

```
<!DOCTYPE name SYSTEM "address.dtd" [...]>
```

Public Identifiers

- Public identifiers provide a mechanism to locate DTD resources and is written as follows

```
<!DOCTYPE name PUBLIC "-//Beginning XML//DTD Address Example//EN">
```

- Keyword PUBLIC, followed by a specialized identifier. Public identifiers are used to identify an entry in a catalog. Public identifiers can follow any format, however, a commonly used format is called Formal Public Identifiers, or FPIs.

4.12 XML SCHEMA DOM AND PRESENTING XML

4.12.1 XML Schema

- XML Schema is commonly known as XML Schema Definition (XSD).
- It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types.
- Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

Syntax

Example

The following example shows how to use schema –

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

<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">

    <xs:element name = "contact">

        <xs:complexType>

            <xs:sequence>

                <xs:element name = "name" type = "xs:string" />
                <xs:element name = "company" type = "xs:string" />
                <xs:element name = "phone" type = "xs:int" />

            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

```

</xs:complexType>
</xs:element>
</xs:schema>



- The basic idea behind XML Schemas is that they describe the legitimate format that an XML document can take.

```

Elements

Elements are the building blocks of XML document. An element can be defined within an XSD as follows –

```
<xs:element name = "x" type = "y"/>
```

Definition Types

XML schema can be define elements in the following ways –

Simple Type

Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example –

```
<xs:element name = "phone_number" type = "xs:int" />
```

Complex Type

A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example –

```

<xs:element name = "Address">
<xs:complexType>
<xs:sequence>
<xs:element name = "name" type = "xs:string" />
<xs:element name = "company" type = "xs:string" />
<xs:element name = "phone" type = "xs:int" />
</xs:sequence>
</xs:complexType>
</xs:element>

```

In the above example, *Address* element consists of child elements. This is a container for other `<xs:element>` definitions, that allows to build a simple hierarchy of elements in the XML document.

Global Types

With the global type, you can define a single type in your document, which can be used by all other references. For example, suppose you want to generalize the *person* and *company* for different addresses of the company. In such case, you can define a general type as follows –

```
<xs:element name = "AddressType">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name = "name" type = "xs:string" />  
      <xs:element name = "company" type = "xs:string" />  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>
```

Now use this type in our example as follows –

```
<xs:element name = "Address1">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name = "address" type = "AddressType" />  
      <xs:element name = "phone1" type = "xs:int" />  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
  
<xs:element name = "Address2">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name = "address" type = "AddressType" />  
      <xs:element name = "phone2" type = "xs:int" />  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>
```

```
</xs:sequence>  
</xs:complexType>  
</xs:element>
```

- Instead of having to define the name and the company twice (once for *Address1* and once for *Address2*), we now have a single definition.
- This makes maintenance simpler, i.e., if you decide to add "Postcode" elements to the address, you need to add them at just one place.

Attributes

Attributes in XSD provide extra information within an element. Attributes have *name* and *type* property as shown below –

```
<xs:attribute name = "x" type = "y"/>
```

4.12.2 XML DOM

- The Document Object Model (DOM) is the foundation of XML. XML documents have a hierarchy of informational units called *nodes*; DOM is a way of describing those nodes and the relationships between them.
- A DOM document is a collection of nodes or pieces of information organized in a hierarchy. This hierarchy allows a developer to navigate through the tree looking for specific information. Because it is based on a hierarchy of information, the DOM is said to be *tree based*.
- The XML DOM, on the other hand, also provides an API that allows a developer to add, edit, move, or remove nodes in the tree at any point in order to create an application.

Example

- The following example (sample.htm) parses an XML document ("address.xml") into an XML DOM object and then extracts some information from it with JavaScript –

```
<!DOCTYPE html>  
<html>  
  <body>  
    <h1>TutorialsPoint DOM example </h1>  
    <div>  
      <b>Name:</b> <span id = "name"></span><br>  
      <b>Company:</b> <span id = "company"></span><br>  
      <b>Phone:</b> <span id = "phone"></span>  
    </div>
```

```

<script>

if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp = new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  }

xmlhttp.open("GET","/xml/address.xml",false);
xmlhttp.send();
xmlDoc = xmlhttp.responseXML;
document.getElementById("name").innerHTML=
  xmlDoc.getElementsByTagName("name")[0].childNodes[0].nodeValue;
document.getElementById("company").innerHTML=
  xmlDoc.getElementsByTagName("company")[0].childNodes[0].nodeValue;
document.getElementById("phone").innerHTML=
  xmlDoc.getElementsByTagName("phone")[0].childNodes[0].nodeValue;
</script>

</body>
</html>

```

Contents of address.xml are as follows –

```

<?xml version = "1.0"?>
<contact-info>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</contact-info>

```

4.13 XML PARSER AND VALIDATION

4.13.1 XML Parser

- **XML parser** is a software library or a package that provides interface for client applications to work with XML documents. It checks for proper format of the XML document and may also validate the XML documents. Modern day browsers have built-in XML parsers.
- Following diagram shows how XML parser interacts with XML document –



Fig:4.6 XML Parser

- The goal of a parser is to transform XML into a readable code.
- To ease the process of parsing, some commercial products are available that facilitate the breakdown of XML document and yield more reliable results.
- Some commonly used parsers are listed below –
 - **MSXML (Microsoft Core XML Services)** – This is a standard set of XML tools from Microsoft that includes a parser.
 - **System.Xml.XmlDocument** – This class is part of .NET library, which contains a number of different classes related to working with XML.
 - **Java built-in parser** – The Java library has its own parser. The library is designed such that you can replace the built-in parser with an external implementation such as Xerces from Apache or Saxon.
 - **Saxon** – Saxon offers tools for parsing, transforming, and querying XML.
 - **Xerces** – Xerces is implemented in Java and is developed by the famous open source Apache Software Foundation.

4.13.2 XML Validation

- Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration (DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are –
 - Well-formed XML document
 - Valid XML document
 - Well-formed XML Document
- An XML document is said to be well-formed if it adheres to the following rules –
 - Non DTD XML files must use the predefined character entities for amp(&), apos(single quote), gt(>), lt(<), quot(double quote).
 - It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
 - Each of its opening tags must have a closing tag or it must be a self ending tag. (<title>....</title> or <title/>). It must have only one attribute in a start tag, which needs to be quoted.

- amp(&), apos(single quote), gt(>), lt(<), quot(double quote) entities other than these must be declared.

Example

Following is an example of a well-formed XML document –

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address
[
    <!ELEMENT address (name,company,phone)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT company (#PCDATA)>
    <!ELEMENT phone (#PCDATA)>
]>
<address>
    <name>Tanmay Patil</name>
    <company>TutorialsPoint</company>
    <phone>(011) 123-4567</phone>
</address>
```

The above example is said to be well-formed as –

- It defines the type of document. Here, the document type is element type.
- It includes a root element named as address.
- Each of the child elements among name, company and phone is enclosed in its self explanatory tag.
- Order of the tags is maintained.

Valid XML Document

- If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.

4.14 XSL AND XSLT TRANSFORMATION

- XSL which stands for **EXtensible Stylesheet Language**. It is similar to XML as CSS is to HTML.

Need for XSL

- In case of HTML document, tags are predefined such as table, div, and span; and the browser knows how to add style to them and display those using CSS styles.
- But in case of XML documents, tags are not predefined. In order to understand and style an XML document, World Wide Web Consortium (W3C) developed XSL which can act as XML based Stylesheet Language. An XSL document specifies how a browser should render an XML document.
- Following are the main parts of XSL –
 - ✓ **XSLT** – used to transform XML document into various other types of document.
 - ✓ **XPath** – used to navigate XML document.
 - ✓ **XSL-FO** – used to format XML document.

What is XSLT?

- ✓ XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.

How XSLT Works?

- An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format.
- XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format.
- This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.

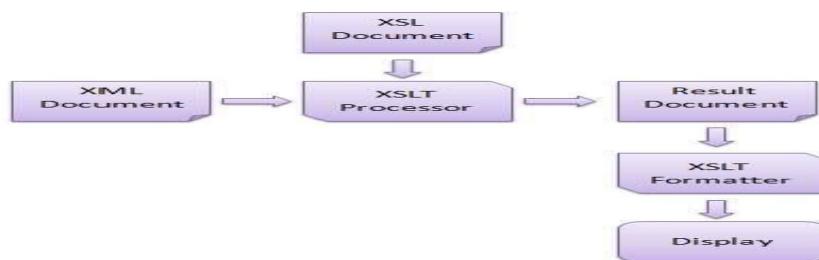


Fig: 4.7 XSLT Transformations

Advantages:

Here are the advantages of using XSLT –

- Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
- Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

4.15 NEWS FEED (RSS AND ATOM)

4.15.1 RSS News feed:

- RSS is the short name of Really Simple Syndication.
- RSS is like a broadcast system which is used to deliver updated information to multiple receivers.
- RSS is used to deliver regular changing web contents to the user.

RSS Feed Basic Structure:

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

<rss version='2.0'>

<channel>

<title>Title of Webpage</title>

<link>Webpage URL</link>

<description>About Webpage</description>

<language>en-us</language>

<item>

<title>Article Title</title>

<link>Article URL</link>

<description>Article Content</description>

</item>

</channel>

</rss>
```

UNIT V

INTRODUCTION TO AJAX and WEB SERVICES

AJAX: Ajax Client Server Architecture-XML Http Request Object-Call Back Methods; **Web Services:** Introduction- Java web services Basics – Creating, Publishing, Testing and Describing a Web services (WSDL)-Consuming a web service, Database Driven web service from an application –SOAP.

TEXT BOOK:

1.Deitel and Deitel and Nieto, —Internet and World Wide Web - How to Programll, Prentice Hall, 5th Edition, 2011.

REFERENCES:

- 1.Stephen Wynkoop and John Burke —Running a Perfect Website ll, QUE, 2nd Edition, 1999.
- 2.Chris Bates, Web Programming – Building Intranet Applications, 3rd Edition, Wiley Publications, 2009.
- 3.Jeffrey C and Jackson, —Web Technologies A Computer Science Perspective ll, Pearson Education, 2011.
- 4.Gopalan N.P. and Akilandeswari J., —Web Technology ll, Prentice Hall of India, 2011.
- 5.UttamK.Roy, —Web Technologies ll, Oxford University Press, 2011.

5.1 AJAX: AJAX CLIENT SERVER ARCHITECTURE

AJAX stands for **A**synchronous **J**ava**S**cript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven.

How AJAX works?

AJAX communicates with the server using XMLHttpRequest object. Let's try to understand the flow of ajax or how ajax works by the image displayed below.

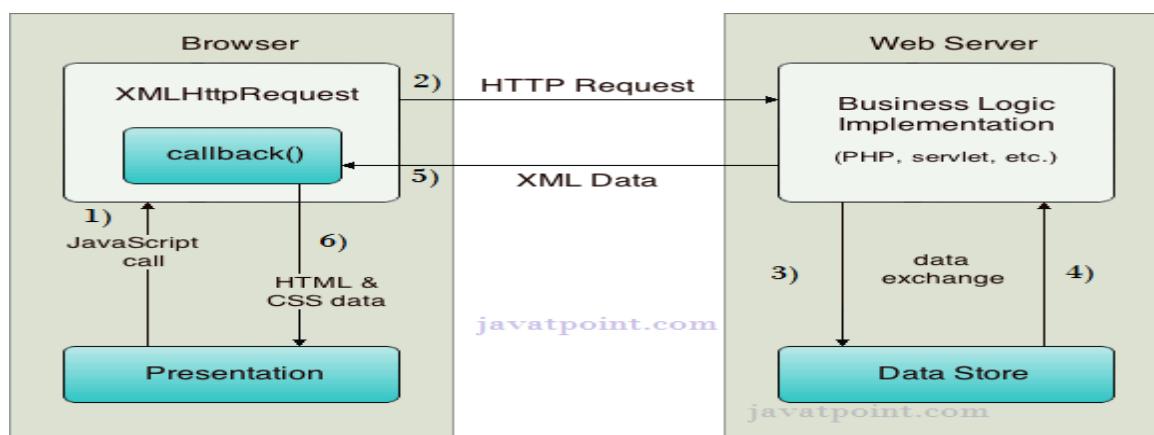


Fig: 5.1 AJAX Architecture

XMLHttpRequest object plays a important role.

1. User sends a request from the UI and a javascript call goes to XMLHttpRequest object.
2. HTTP Request is sent to the server by XMLHttpRequest object.
3. Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.
4. Data is retrieved.
5. Server sends XML data or JSON data to the XMLHttpRequest callback function.
6. HTML and CSS data is displayed on the browser.

5.2 XML HTTP REQUEST OBJECT

- The XMLHttpRequest object is the key to AJAX. It has been available ever since Internet Explorer 5.5 was released in July 2000, but was not fully discovered until AJAX and Web 2.0 in 2005 became popular.
- XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.
- The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.
- Listed below are some of the methods and properties that you have to get familiar with.

5.2.1 XMLHttpRequest Methods

- **abort()**-Cancels the current request.
- **getAllResponseHeaders()**-Returns the complete set of HTTP headers as a string.
- **getResponseHeader(headerName)**-Returns the value of the specified HTTP header.
- **open(method, URL)**
- **open(method, URL, async)**
- **open(method, URL, async, userName)**
- **open(method, URL, async, userName, password)**
- Specifies the method, URL, and other optional attributes of a request.
- The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.
- The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without

waiting for a response, and "false" means that the script waits for a response before continuing script processing.

- **send(content)**-Sends the request.
- **setRequestHeader(label, value)**-Adds a label/value pair to the HTTP header to be sent.

5.2.3 XMLHttpRequest Properties

- **onreadystatechange**-An event handler for an event that fires at every state change.
- **readyState**-The readyState property defines the current state of the XMLHttpRequest object.

The following table provides a list of the possible values for the readyState property –

State	Description
0	The request is not initialized.
1	The request has been set up.
2	The request has been sent.
3	The request is in process.
4	The request is completed.

- **readyState = 0** After you have created the XMLHttpRequest object, but before you have called the open() method.
- **readyState = 1** After you have called the open() method, but before you have called send().
- **readyState = 2** After you have called send().
- **readyState = 3** After the browser has established a communication with the server, but before the server has completed the response.
- **readyState = 4** After the request has been completed, and the response data has been completely received from the server.
- **responseText**-Returns the response as a string.
- **responseXML**-Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.
- **Status**-Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
- **Status Text**-Returns the status as a string (e.g., "Not Found" or "OK").

5.3 CALL BACK METHODS

The ajaxSuccess(callback) method attaches a function to be executed whenever an AJAX request completes successfully. This is an Ajax Event.

Syntax

Here is the simple syntax to use this method –

```
$(document).ajaxSuccess( callback )
```

Parameters

Here is the description of all the parameters used by this method –

- **callback** – The function to execute. The event object, XMLHttpRequest, and settings used for that request are passed as arguments to the callback.

Example (Refer this program for AJAX ARCHITECTURE AND XMLHttpRequest Object)

```
<!DOCTYPE html>

<html>
<body>

<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML =
            this.responseText;
        }
    };
    xhttp.open("GET", "ajax_info.txt", true);
}
```

```
xhttp.send();  
}  
</script>
```

```
</body>
```

```
</html>
```

Output:

The XMLHttpRequest Object

Change Content

After you clicked Change Content

AJAX

AJAX is not a programming language.

AJAX is a technique for accessing web servers from a web page.

AJAX stands for Asynchronous JavaScript And XML.

5.4 WEB SERVICES-INTRODUCTION

- A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language—Java can talk with Perl; Windows applications can talk with Unix applications.
- Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.

- A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

To summarize, a complete web service is, therefore, any service that –

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

Components of Web Services

The basic web services platform is XML + HTTP. All the standard web services work using the following components –

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

How Does a Web Service Work?

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of –

- XML to tag the data
- SOAP to transfer a message
- WSDL to describe the availability of service.

Example

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.

The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.

The steps to perform this operation are as follows –

- The client program bundles the account registration information into a SOAP message.
- This SOAP message is sent to the web service as the body of an HTTP POST request.
- The web service unpacks the SOAP request and converts it into a command that the application can understand.
- The application processes the information as required and responds with a new unique account number for that customer.
- Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
- The client program unpacks the SOAP message to obtain the results of the account registration process.

5.5 WSDL

WSDL is an XML-based file which basically tells the client application what the web service does. It is known as the Web Services Description Language(WSDL).

The WSDL file is used to describe in a nutshell what the web service does and gives the client all the information required to connect to the web service and use all the functionality provided by the web service.

5.5.1 WSDL Elements

The WSDL file contains the following main parts

1. The **<types>** tag is used to define all the complex datatypes, which will be used in the message exchanged between the client application and the web service. This is an important aspect of the client application, because if the web service works with a complex data type, then the client application should know how to process the complex data type. Data types such as float, numbers, and strings are all simple data types, but there could be structured data types which may be provided by the web service.

For example, there could be a data type called EmployeeDataType which could have 2 elements called "EmployeeName" of type string and "EmployeeID" of type number or integer. Together they form a data structure which then becomes a complex data type.

2. The **<messages>** tag is used to define the message which is exchanged between the client application and the web server. These messages will explain the input and output operations which can be performed by the web service. An example of a message can be a message which accepts the EmployeeID of an employee, and the output message can be the name of the employee based on the EmployeeID provided.

3. The **<portType>** tag is used to encapsulate every input and output message into one logical operation. So there could be an operation called "GetEmployee" which combines the input message of accepting the EmployeeID from a client application and then sending the EmployeeName as the output message.
4. The **<binding>** tag is used to bind the operation to the particular port type. This is so that when the client application calls the relevant port type, it will then be able to access the operations which are bound to this port type. Port types are just like interfaces. So if a client application needs to use a web service they need to use the binding information to ensure that they can connect to the interface provided by that web service.
5. The **<service>** tag is a name given to the web service itself. Initially, when a client application makes a call to the web service, it will do by calling the name of the web service. For example, a web service can be located at an address such as **http://localhost/Guru99/Tutorial.asmx**. The service tag will actually have the URL defined as **http://localhost/Guru99/Tutorial.asmx**, which will actually tell the client application that there is a web service available at this location.

5.5.2 WSDL Message Part

- The WSDL consists of a section called "messages" which is denoted by the **<message>** element.
- This element is basically used to describe the data that gets exchanged between the web service and the client application.
- Each web service will always have 2 types of messages,
 - One is for the input of the web service, and the other is for the output of the web service.
 - The input is used to describe the parameters which are accepted by the web service. This is an important aspect of the client application so that it knows the values to be sent as parameters to the web service.
 - The other type of message is the output message which tells what results are provided by the web service.
- Each message, in turn, will have a **<part>** element which is used to describe the parameter used by the input and output message.
- Below is a simple example, of what a message for a web service looks like. The functionality of the web service is to provide the name of a "Tutorial" once a "Tutorial ID" is submitted as a parameter to the web service.



Fig: 5.2 WSDL Message Part

1. As we can see the web service has 2 messages, one for the input and the other for the output.
2. The input message is known as TutorialNameRequest which has one parameter called TutorialID. This parameter is of the type number which is specified by the xsd:number type
3. The output message is known as TutorialNameResponse which has one parameter called TutorialName. This parameter is of the type string which is specified by the xsd:string type

5.5.3 Creating WSDL File

- The WSDL file gets created whenever a web service is built in any programming language.
- Since the WSDL file is pretty complicated to be generated from plain scratch, all editors such as Visual Studio for .Net and Eclipse for Java automatically create the WSDL file.
- Below is an example of a WSDL file created in Visual Studio.

```

<?xml version="1.0"?>
<definitions name="Tutorial"
             targetNamespace=http://Guru99.com/Tutorial.wsdl
             xmlns:tns=http://Guru99.com/Tutorial.wsdl
             xmlns:xsd1=http://Guru99.com/Tutorial.xsd
             xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
             xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
    <schema targetNamespace=http://Guru99.com/Tutorial.xsd
           xmlns="http://www.w3.org/2000/10/XMLSchema">

        <element name="TutorialNameRequest">
            <complexType>

```

```

        <all>
            <element name="TutorialName" type="string"/>
        </all>
    </complexType>
</element>
<element name="TutorialIDRequest">
    <complexType>
        <all>
            <element name="TutorialID" type="number"/>
        </all>
    </complexType>
</element>
</schema>
</types>
<message name="GetTutorialNameInput">
    <part name="body" element="xsd1:TutorialIDRequest"/>
</message>
<message name="GetTutorialNameOutput">
    <part name="body" element="xsd1:TemplateNameRequest"/>
</message>
<portType name="TutorialPortType">
    <operation name="GetTutorialName">
        <input message="tns:GetTutorialNameInput"/>
        <output message="tns:GetTutorialNameOutput"/>
    </operation>
</portType>
<binding name="TutorialSoapBinding" type="tns:TutorialPortType">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http">
        <operation name="GetTutorialName">
            <soap:operation
soapAction="http://Guru99.com/GetTutorialName"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
</binding>

<service name="TutorialService">
    <documentation>TutorialService</documentation>

```

```

<port name="TutorialPort" binding="tns:TutorialSoapBinding">
    <soap:address location="http://Guru99.com/Tutorial"/>
</port>
</service>
</definitions>

```

5.5.4 Publishing the Web Service Example

- An example of how we can publish a web service and consume it by using Visual Studio.
- **Step 1)** The first step is to create your web service. The detailed steps of how the [Asp.Net](#) web project and a web service is created. The key part is to enter the below code in the Web services file.

```

namespace webservic asmx
{
    [WebService(Name = "Guru99 Web service")]
    public class TutorialService : System.Web.Services.WebService
    {
        [WebMethod]
        public string GetTutorialService(int TutorialID)
        {
            string TutorialName = "Web Services";
            return TutorialName;
        }
    }
}

```

```

namespace webservic.asmx
{
    [WebService(Name = "Guru99 Web service")]
    public class TutorialService : System.Web.Services.WebService
    {
        [WebMethod]
        public string GetTutorialService(int TutorialID)
        {
            string TutorialName = "Web Services";
            return TutorialName;
        }
    }
}

```

Fig: 5.3 creating web Service

Code Explanation:

1. Here we are creating a WebMethod called "Guru99WebService." In this web method, we are including an integer parameter which needs to be passed whenever this web method is called.

2. Next we are defining a variable called "TutorialName" which will hold the string value of "Web Services." This is the value which will be returned when the web service is called.

Step 2) Once we have defined the web services file, the next step is to create a client project which will consume this web service.

- Let's create a simple console application which will call this web service, invoke the "Guru99WebService" and then display the output of the web method in the console log screen. Follow the below steps to create a console application.
- Right-click the Visual Studio solution file and choose the option Add->New project

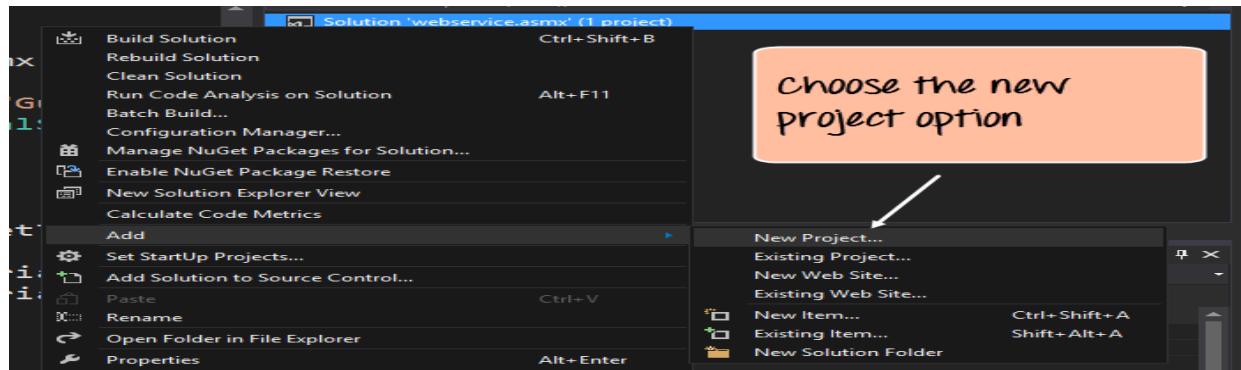


Fig 5.4 Creating a console Application

Step3) In this step,

1. Ensure to first choose the Visual C# Windows option. Then choose the option of creating a console application.
2. Give a name for your project which in our case has been given as "DemoApplication."

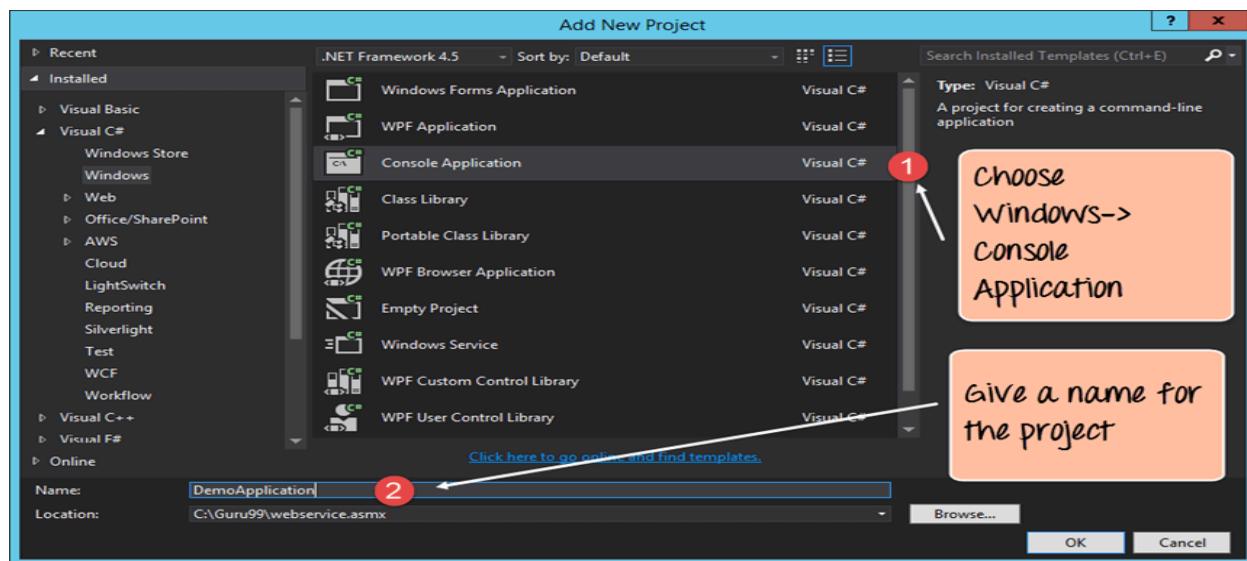


Fig: 5.5 Naming the Project

- After you click the OK button in the above screen, you will be able to see the project in the Solution explorer in Visual Studio.

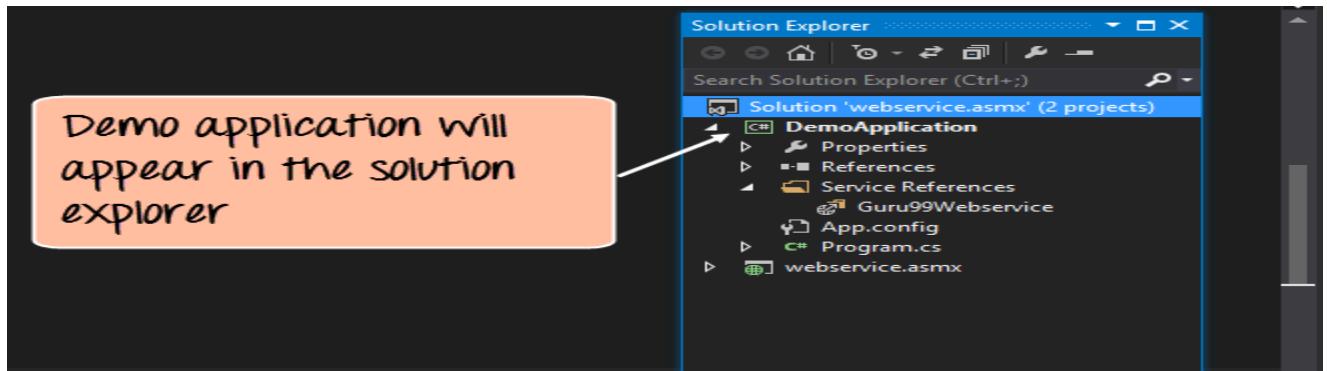


Fig: 5.6 Displaying the Application

Step 4) In this step, you be setting the DemoApplication Console application as the startup project. This is done to ensure that this application launches first when the entire Visual Studio project is run. This Console application will, in turn, call the web service which will be automatically launched by Visual Studio.

- To complete this step, right-click the DemoApplication project and choose the option "Set as StartUp Project."

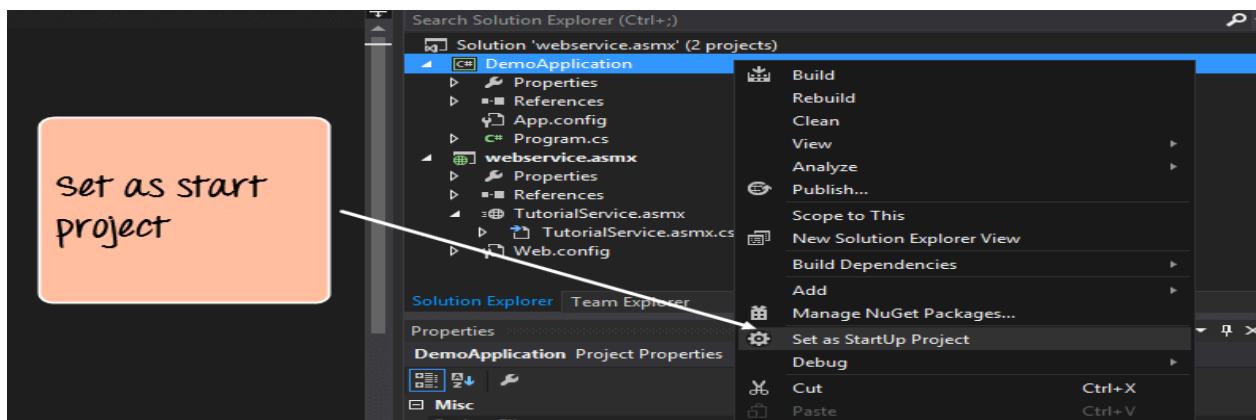


Fig: 5.7 Start the New Project

Step 5) The next step is to add the service reference of our "Guru99Webservice" to our console application. This is done so that the DemoApplication can reference the web service and all of the web methods in the web service.

- To do this, right-click the DemoApplication project file and choose the menu option Add->Service Reference.

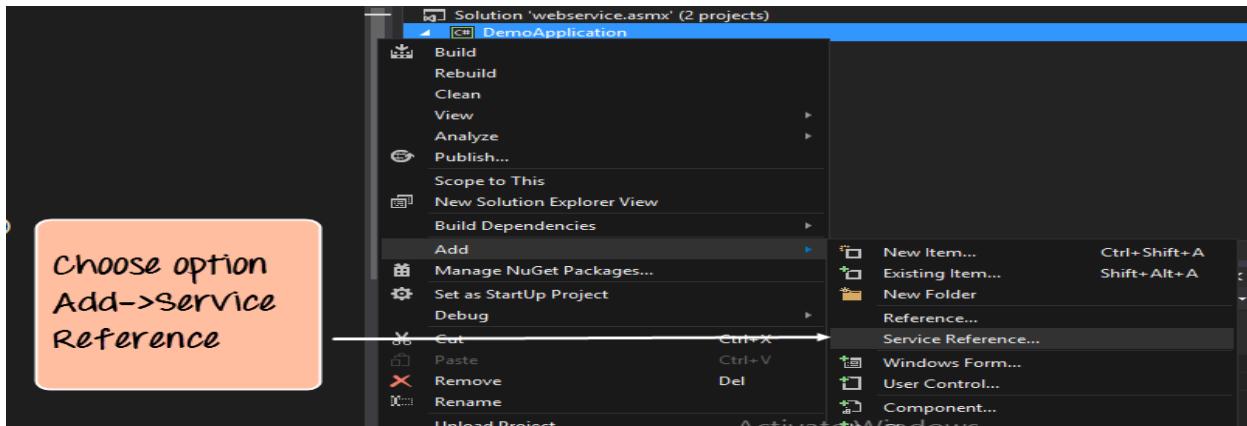


Fig: 5.8 Add the Service Reference

Step 6) In this step, we will provide the different values which are required to add our service reference

1. Firstly we need to choose our discover option. This option will automatically pick up the WSDL file for our TutorialService web service.
2. Next, we should give a name for our service reference. In our case, we are giving it a name of Guru99Webservice.
3. Then we need to expand on the TutorialService.asmx option so that we can have the ability to see the 'GetTutorialService' method on the right-hand side. Here TutorialService.asmx is the name of our Visual Studio .Net file which contains the code for our web service.
4. We will then see our Web method which we had in our web service known as "GetTutorialService"

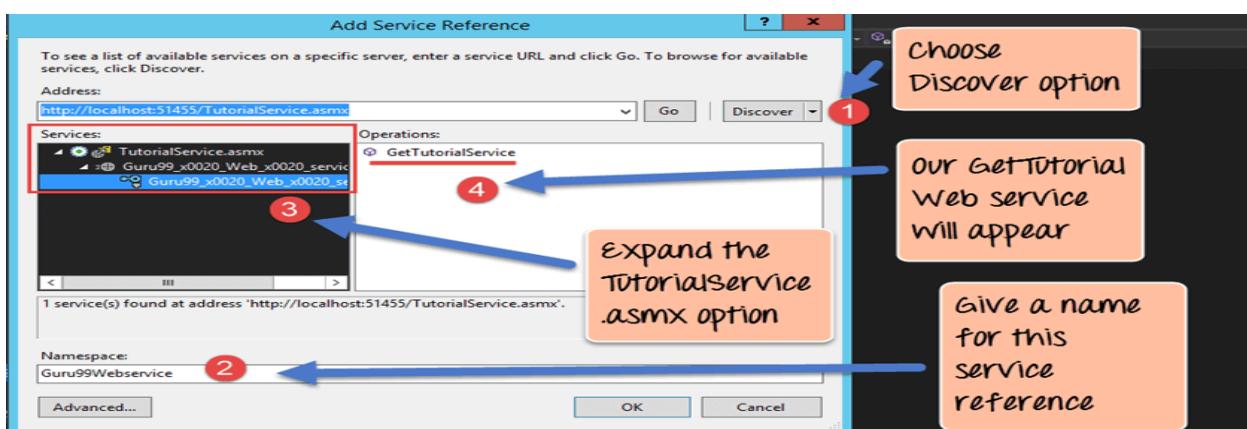


Fig: 5.9 Displaying the Web Service Reference

- When we click on the 'OK' button, all of the required code to access this web service will be added to our DemoApplication Console application as shown below.
- The screenshot shows that the "Guru99Webservice" was successfully added to our console application.

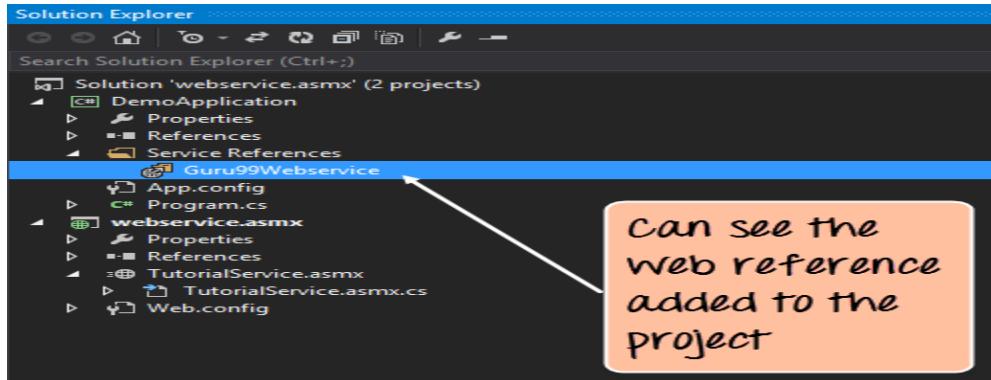


Fig: 5.10 Web reference added to Project

Step 7) The next step is to add the code to our console application to access the web method in our web service. Open the Program.cs code file which comes automatically with the console application and add the below code

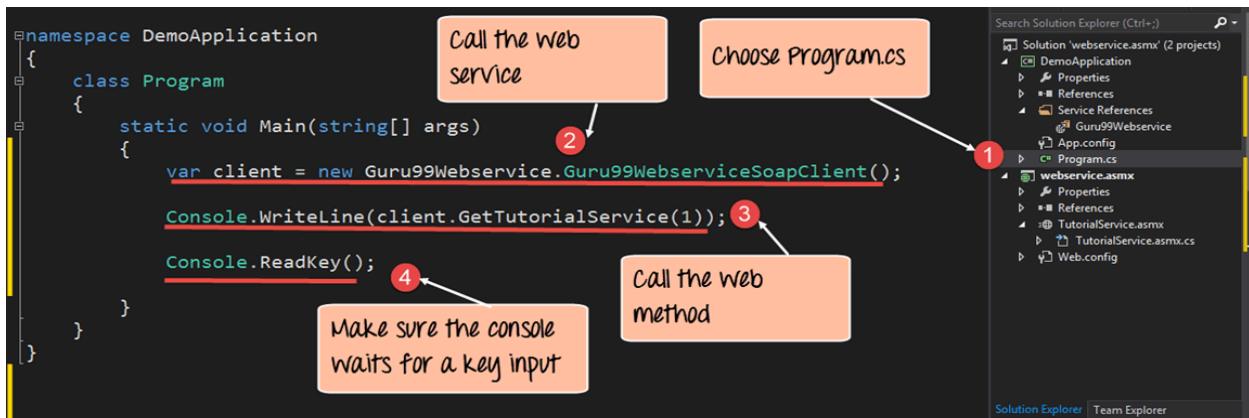


Fig: 5.11 Call the web service

```
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            var client = new Guru99Webservice.Guru99WebserviceSoapClient();
```

```

        Console.WriteLine(client.GetTutorialService(1));

        Console.ReadKey();
    }
}
}

```

Code Explanation:-

1. The first part is to choose the Program.cs file. This is the main file which is created by Visual Studio when a console application is created. This file is what gets executed when the console application (in our case demo application) is executed.
2. We then create a variable called "client" which will be set to an instance of our Service reference which was created in an earlier step. In our case, the service reference is 'Guru99Webservice.Guru99WebserviceSoapClient()'
3. We are then calling our Webmethod 'GetTutorialService' in the TutorialService web service Remember that our GetTutorialService' method accepts an integer parameter, so we are just passing an integer parameter to the web method.
4. This final line is just to ensure the console log screen remains active so that we can view the output. This command will just wait for some input from the user.

Output

- When all of the above steps are followed, and the DemoApplication is run the below output will be displayed.

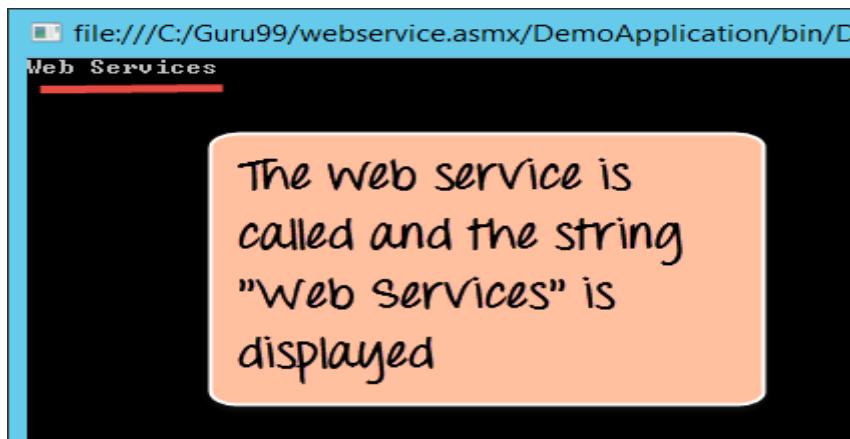


Fig: 5.11 WSDL Output

- From the output, we can clearly see that the DemoApplication calls our Web service and that the string returned by the Web service is displayed in our Console log.

5.6 SOAP

SOAP is an acronym for Simple Object Access Protocol. It is an XML-based messaging protocol for exchanging information among computers. SOAP is an application of the XML specification.

- SOAP is a communication protocol designed to communicate via Internet.
- SOAP can extend HTTP for XML messaging.
- SOAP provides data transport for Web services.
- SOAP can exchange complete documents or call a remote procedure.
- SOAP can be used for broadcasting a message.
- SOAP is platform- and language-independent.
- SOAP is the XML way of defining what information is sent and how.
- SOAP enables client applications to easily connect to remote services and invoke remote methods.
- Although SOAP can be used in a variety of messaging systems and can be delivered via a variety of transport protocols, the initial focus of SOAP is remote procedure calls transported via HTTP.
- Other frameworks including CORBA, DCOM, and Java RMI provide similar functionality to SOAP, but SOAP messages are written entirely in XML and are therefore uniquely platform- and language-independent.

5.6.1 SOAP Message Structure

A SOAP message is an ordinary XML document containing the following elements –

- Envelope – Defines the start and the end of the message. It is a mandatory element.
- Header – Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end-point. It is an optional element.
- Body – Contains the XML data comprising the message being sent. It is a mandatory element.
- Fault – An optional Fault element that provides information about errors that occur while processing the message.
- The following block depicts the general structure of a SOAP message –

```

<?xml version = "1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV =
"http://www.w3.org/2001/12/soap-envelope"
    SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">

    <SOAP-ENV:Header>
        ...
        ...
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        ...
        ...
        <SOAP-ENV:Fault>
            ...
            ...
        </SOAP-ENV:Fault>
        ...
    </SOAP-ENV:Body>
</SOAP_ENV:Envelope>

```

5.6.2 SOAP-Envelope

The SOAP envelope indicates the start and the end of the message so that the receiver knows when an entire message has been received. The SOAP envelope solves the problem of knowing when you are done receiving a message and are ready to process it. The SOAP envelope is therefore basically a packaging mechanism.

- Every SOAP message has a root Envelope element.
- Envelope is a mandatory part of SOAP message.
- Every Envelope element must contain exactly one Body element.
- If an Envelope contains a Header element, it must contain no more than one, and it must appear as the first child of the Envelope, before the Body.
- The envelope changes when SOAP versions change.
- The SOAP envelope is specified using the *ENV* namespace prefix and the Envelope element.
- The optional SOAP encoding is also specified using a namespace name and the optional *encodingStyle* element, which could also point to an encoding style other than the SOAP one.
- A v1.1-compliant SOAP processor generates a fault upon receiving a message containing the v1.2 envelope namespace.
- A v1.2-compliant SOAP processor generates a *VersionMismatch* fault if it receives a message that does not include the v1.2 envelope namespace.

V1.2-Compliant SOAP Message

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
    SOAP-ENV:encodingStyle = " http://www.w3.org/2001/12/soap-encoding">
    ...
    Message information goes here
    ...
</SOAP-ENV:Envelope>
```

SOAP with HTTP POST

- o The following example illustrates the use of a SOAP message within an HTTP POST operation, which sends the message to the server.
- o It shows the namespaces for the envelope schema definition and for the schema definition of the encoding rules. T
- o The OrderEntry reference in the HTTP header is the name of the program to be invoked at the tutorialspoint.com website.

POST /OrderEntry HTTP/1.1
Host: www.tutorialspoint.com
Content-Type: application/soap; charset="utf-8"
Content-Length: nnnn

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
    SOAP-ENV:encodingStyle = " http://www.w3.org/2001/12/soap-encoding">
    ...
    Message information goes here
    ...
</SOAP-ENV:Envelope>
```

5.6.3 SOAP-Header

- The optional Header element offers a flexible framework for specifying additional application-level requirements. For example, the Header element can be used to specify a digital signature for password-protected services. Likewise, it can be used to specify an account number for pay-per-use SOAP services.
- It is an optional part of a SOAP message.
- Header elements can occur multiple times.
- Headers are intended to add new features and functionality.
- The SOAP header contains header entries defined in a namespace.

- The header is encoded as the first immediate child element of the SOAP envelope.
- When multiple headers are defined, all immediate child elements of the SOAP header are interpreted as SOAP header blocks.

SOAP Header Attributes

A SOAP Header can have the following two attributes –

Actor attribute

- The SOAP protocol defines a message path as a list of SOAP service nodes.
- Each of these intermediate nodes can perform some processing and then forward the message to the next node in the chain.
- By setting the Actor attribute, the client can specify the recipient of the SOAP header.

MustUnderstand attribute

- It indicates whether a Header element is optional or mandatory. If set to true, the recipient must understand and process the Header attribute according to its defined semantics, or return a fault.
- The following example shows how to use a Header in a SOAP message.

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = " http://www.w3.org/2001/12/soap-envelope"
    SOAP-ENV:encodingStyle = "
http://www.w3.org/2001/12/soap-encoding">

    <SOAP-ENV:Header>
        <t:Transaction
            xmlns:t = "http://www.tutorialspoint.com/transaction/"
            SOAP-ENV:mustUnderstand = "true">5
        </t:Transaction>
    </SOAP-ENV:Header>
    ...
    ...
</SOAP-ENV:Envelope>
```

5.6.4 SOAP-Body

- The SOAP body is a mandatory element that contains the application-defined XML data being exchanged in the SOAP message. The body must be contained within the envelope and must follow any headers that might be defined for the message.
- The body is defined as a child element of the envelope, and the semantics for the body are defined in the associated SOAP schema.

- The body contains mandatory information intended for the ultimate receiver of the message. For example –

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope>
    .....
    <SOAP-ENV:Body>
        <m:GetQuotation xmlns:m = "http://www.tp.com/Quotation">
            <m:Item>Computers</m:Item>
        </m:GetQuotation>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- The example above requests a quotation of computer sets. Note that the m:GetQuotation and the Item elements above are application-specific elements. They are not a part of the SOAP standard.
- Here is the response to the above query –

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope>
    .....
    <SOAP-ENV:Body>
        <m:GetQuotationResponse xmlns:m = "http://www.tp.com/Quotation">
            <m:Quotation>This is Quotation</m:Quotation>
        </m:GetQuotationResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- Normally, the application also defines a schema to contain semantics associated with the request and response elements.
- The *Quotation* service might be implemented using an EJB running in an application server; if so, the SOAP processor would be responsible for mapping the body information as parameters into and out of the EJB implementation of the *GetQuotationResponse* service. The SOAP processor could also be mapping the body information to a .NET object, a CORBA object, a COBOL program, and so on.

5.6.5 SOAP-Fault

- If an error occurs during processing, the response to a SOAP message is a SOAP fault element in the body of the message, and the fault is returned to the sender of the SOAP message.

- The SOAP fault mechanism returns specific information about the error, including a predefined code, a description, and the address of the SOAP processor that generated the fault.
- A SOAP message can carry only one fault block.
- Fault is an optional part of a SOAP message.
- For HTTP binding, a successful response is linked to the 200 to 299 range of status codes.
- SOAP Fault is linked to the 500 to 599 range of status codes.

Sub-elements of Fault

The SOAP Fault has the following sub elements –

Sr.No	Sub-element & Description
1	<faultCode> It is a text code used to indicate a class of errors. See the next Table for a listing of predefined fault codes.
2	<faultString> It is a text message explaining the error.
3	<faultActor> It is a text string indicating who caused the fault. It is useful if the SOAP message travels through several nodes in the SOAP message path, and the client needs to know which node caused the error. A node that does not act as the ultimate destination must include a faultActor element.
4	<detail> It is an element used to carry application-specific error messages. The detail element can contain child elements called detail entries.

SOAP Fault Codes

The faultCode values defined below must be used in the *faultcode* element while describing faults.

Sr.No	Error & Description
1	SOAP-ENV:VersionMismatch Found an invalid namespace for the SOAP Envelope element.
2	SOAP-ENV:MustUnderstand An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood.
3	SOAP-ENV:Client The message was incorrectly formed or contained incorrect information.
4	SOAP-ENV:Server

There was a problem with the server, so the message could not proceed

SOAP Fault Example

The following code is a sample Fault. The client has requested a method named *ValidateCreditCard*, but the service does not support such a method. This represents a client request error, and the server returns the following SOAP response –

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode xsi:type = "xsd:string">SOAP-ENV:Client</faultcode>
            <faultstring xsi:type = "xsd:string">
                Failed to locate method (ValidateCreditCard) in class (examplesCreditCard) at
                /usr/local/ActivePerl-5.6/lib/site_perl/5.6.0/SOAP/Lite.pm line 1555.
            </faultstring>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5.6.7 SOAP-Encoding

SOAP includes a built-in set of rules for encoding data types. It enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

- SOAP data types are divided into two broad categories – scalar types and compound types.
- Scalar types contain exactly one value such as a last name, price, or product description.
- Compound types contain multiple values such as a purchase order or a list of stock quotes.
- Compound types are further subdivided into arrays and structs.
- The encoding style for a SOAP message is set via the *SOAP-ENV:encodingStyle* attribute.
- To use SOAP 1.1 encoding, use the value <http://schemas.xmlsoap.org/soap/encoding/>

- To use SOAP 1.2 encoding, use the value <http://www.w3.org/2001/12/soap-encoding>
- Latest SOAP specification adopts all the built-in types defined by XML Schema. Still, SOAP maintains its own convention for defining constructs not standardized by XML Schema, such as arrays and references.

Scalar Types

- For scalar types, SOAP adopts all the built-in simple types specified by the XML Schema specification. This includes strings, floats, doubles, and integers.
- The following table lists the main simple types, excerpted from the XML Schema Part 0 – Primer <http://www.w3.org/TR/2000/WD-xmlschema-0-20000407/>
- For example, here is a SOAP response with a double data type –

Simple Types Built-In to XML Schema	
Simple Type	Example
string	Confirm this is electric.
boolean	true, false, 1, 0.
float	-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN.
double	-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN.
decimal	-1.23, 0, 123.4, 1000.00.
binary	100010
integer	-126789, -1, 0, 1, 126789.

```

<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:getPriceResponse
            xmlns:ns1 = "urn:examples:priceservice">

```

```

SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding"
    <return xsi:type = "xsd:double">54.99</return>
</ns1:getPriceResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Compound Types

- SOAP arrays have a very specific set of rules, which require that you specify both the element type and array size. SOAP also supports multidimensional arrays, but not all SOAP implementations support multidimensional functionality.
- To create an array, you must specify it as an *xsi:type* of array. The array must also include an *arrayType* attribute. This attribute is required to specify the data type for the contained elements and the dimension(s) of the array.
- For example, the following attribute specifies an array of 10 double values –

arrayType = "xsd:double[10]"

- In contrast, the following attribute specifies a two-dimensional array of strings –

arrayType = "xsd:string[5,5]"

- Here is a sample SOAP response with an array of double values –

```

<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:getPriceListResponse
            xmlns:ns1 = "urn:examples:pricelistservice"
            SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">

            <return xmlns:ns2 = "http://www.w3.org/2001/09/soap-encoding"
                xsi:type = "ns2:Array" ns2:arrayType = "xsd:double[2]">
                <item xsi:type = "xsd:double">54.99</item>
                <item xsi:type = "xsd:double">19.99</item>
            </return>
        </ns1:getPriceListResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

- Structs contain multiple values, but each element is specified with a unique accessor element. For example, consider an item within a product catalog. In this case, the struct might contain a product SKU, product name, description, and price. Here is how such a struct would be represented in a SOAP message –

```

<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:getProductResponse
            xmlns:ns1 = "urn:examples:productservice"
            SOAP-ENV:encodingStyle =
            "http://www.w3.org/2001/12/soap-encoding">

            <return xmlns:ns2 = "urn:examples" xsi:type = "ns2:product">
                <name xsi:type = "xsd:string">Red Hat Linux</name>
                <price xsi:type = "xsd:double">54.99</price>
                <description xsi:type = "xsd:string">
                    Red Hat Linux Operating System
                </description>
                <SKU xsi:type = "xsd:string">A358185</SKU>
            </return>
        </ns1:getProductResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5.6.8 SOAP-Transport

- SOAP is not tied to any transport protocol. SOAP can be transported via SMTP, FTP, IBM's MQSeries, or Microsoft Message Queuing (MSMQ).
- SOAP specification includes details on HTTP only. HTTP remains the most popular SOAP transport protocol.

SOAP via HTTP

- Quite logically, SOAP requests are sent via an HTTP request and SOAP responses are returned within the content of the HTTP response. While SOAP requests can be sent via an HTTP GET, the specification includes details on HTTP POST only.
- Additionally, both HTTP requests and responses are required to set their content type to text/xml.

- The SOAP specification mandates that the client must provide a *SOAPAction header*, but the actual value of the SOAPAction header is dependent on the SOAP server implementation.
- For example, to access the AltaVista BabelFish Translation service, hosted by XMethods, you must specify the following as a SOAPAction header.

urn:xmethodsBabelFish#BabelFish

- Even if the server does not require a full SOAPAction header, the client must specify an empty string ("") or a null value. For example –

SOAPAction: ""
SOAPAction:

- Here is a sample request sent via HTTP to the XMethods Babelfish Translation service –

```
POST /perl/soaplite.cgi HTTP/1.0
Host: services.xmethods.com
Content-Type: text/xml; charset = utf-8
Content-Length: 538
SOAPAction: "urn:xmethodsBabelFish#BabelFish"

<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:BabelFish
            xmlns:ns1 = "urn:xmethodsBabelFish"
            SOAP-ENV:encodingStyle =
"http://schemas.xmlsoap.org/soap/encoding/">
            <translationmode xsi:type =
"xsd:string">en_fr</translationmode>
            <sourcedata xsi:type = "xsd:string">Hello,
world!</sourcedata>
        </ns1:BabelFish>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- Note the content type and the SOAPAction header. Also note that the BabelFish method requires two String parameters. The translation mode en_fr translates from English to French. Here is the response from XMethods –

```

HTTP/1.1 200 OK
Date: Sat, 09 Jun 2001 15:01:55 GMT
Server: Apache/1.3.14 (Unix) tomcat/1.0 PHP/4.0.1pl2
SOAPServer: SOAP::Lite/Perl/0.50
Cache-Control: s-maxage = 60, proxy-revalidate
Content-Length: 539
Content-Type: text/xml

<?xml version = "1.0" encoding = "UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENC = "http://schemas.xmlsoap.org/soap/encoding/"
    SOAP-ENV:encodingStyle =
"http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"
    xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

    <SOAP-ENV:Body>
        <namesp1:BabelFishResponse xmlns:namesp1 =
"urn:xmethodsBabelFish">
            <return xsi:type = "xsd:string">Bonjour, monde!</return>
        </namesp1:BabelFishResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

- SOAP responses delivered via HTTP are required to follow the same HTTP status codes. For example, a status code of 200 OK indicates a successful response. A status code of 500 Internal Server Error indicates that there is a server error and that the SOAP response includes a Fault element.

Note: Database Driven web service from an application

For the above topic refer following link

https://www.brainkart.com/article/Consuming-a-Database-Driven-Web-Service-from-a--Web-Application_11074/

UNIT I WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0**9**

Web Essentials: Clients, Servers and Communication – The Internet – Basic Internet protocols – World wide web – HTTP Request Message – HTTP Response Message – Web Clients – Web Servers – HTML5 – Tables – Lists – Image – HTML5 control elements – Semantic elements – Drag and Drop – Audio – Video controls – CSS3 – Inline, embedded and external style sheets – Rule cascading – Inheritance – Backgrounds – Border Images – Colors – Shadows – Text – Transformations – Transitions – Animations.

WEB ESSENTIALS

1.1 Web Essentials:

Server:

The software that distributes the information and the machine where the information and software reside is called the server.

- provides requested service to client
- e.g., Web server sends requested Web page

Client:

The software that resides on the remote machine, communicates with the server, fetches the information, processes it, and then displays it on the remote machine is called the client.

- initiates contact with server (—speaks first!)
- typically requests service from server
- Web: client implemented in browser

Web server:

Software that delivers Web pages and other documents to browsers using the HTTP protocol

Web Page:

A web page is a document or resource of information that is suitable for the World Wide Web and can be accessed through a web browser.

Website:

A collection of pages on the World Wide Web that are accessible from the same URL and typically residing on the same server.

URL:

Uniform Resource Locator, the unique address which identifies a resource on the Internet for routing purposes.

1.2 Client-server paradigm:

The Client-Server paradigm is the most prevalent model for distributed computing protocols. It is the basis of all distributed computing paradigms at a higher level of abstraction. It is service-oriented, and employs a request-response protocol.

A server process, running on a server host, provides access to a service. A client process, running on a client host, accesses the service via the server process. The interaction of the process proceeds according to a protocol.

The primary idea of a client/server system is that you have a central repository of information—some kind of data, often in a database—that you want to distribute on demand to some set of people or machines.

1.3 The Internet:

- Medium for communication and interaction in inter connected network.
- Makes information constantly and instantly available to anyone with a connection.

Web Browsers:

- User agent for Web is called a browser:
 - o Internet Explorer

- o Firefox

Web Server:

- Server for Web is called Web server:

- o Apache (public domain)

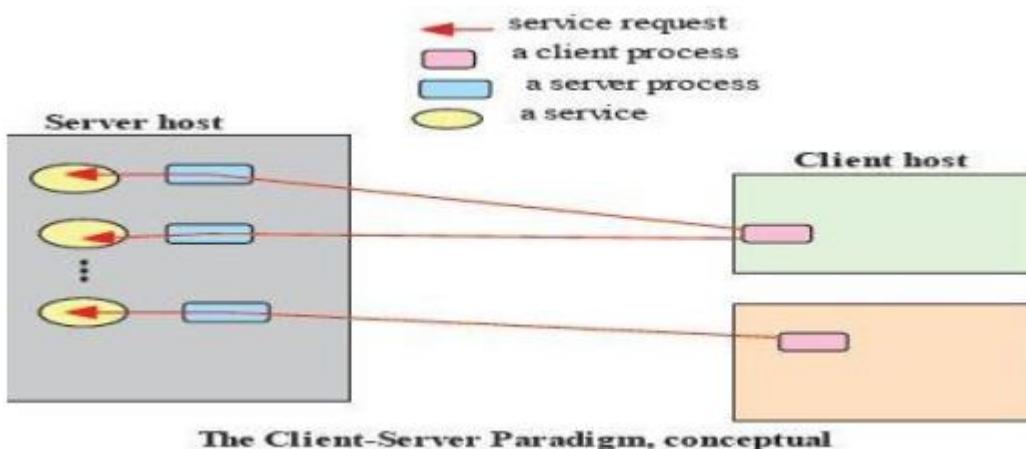
- o MS Internet Information Server

Protocol:

Protocols are agreed formats for transmitting data between devices.

The protocol determines:

- i. The error checking required
- ii. Data compression method used
- iii. The way the end of a message is signalled
- iv. The way the device indicates that it has received the message



1.4 Internet Protocol:

There are many protocols used by the Internet and the WWW:

- o TCP/IP
- o HTTP
- o FTP
- o Electronic mail protocols IMAP
- o POP

TCP/IP

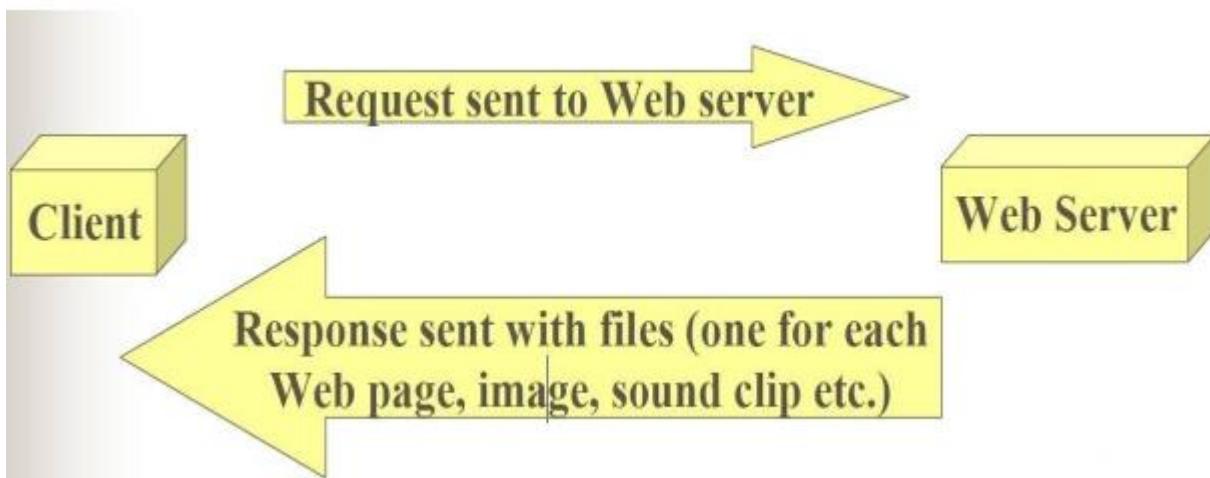
The Internet uses two main protocols (developed by Vincent Cerf and Robert Kahn)

Transmission control protocol (TCP):Controls disassembly of message into packets at the origin reassembles at the destination

Internet protocol (IP):Specifies the addressing details for each packet Each packet is labelled with its origin and destination.

1.5 Hypertext Transfer Protocol (HTTP)

- The hypertext transfer protocol (HTTP) was developed by Tim Berners-Lee in 1991
- HTTP was designed to transfer pages between machines
- The client (or Web browser) makes a request for a given page and the Server is responsible for finding it and returning it to the client
- The browser connects and requests a page from the server
- The server reads the page from the file system, sends it to the client and terminates the connection.



Electronic Mail Protocols:

- Electronic mail uses the client/server model
- The organisation has an email server devoted to handling email
 - Stores and forwards email messages
 - Individuals use email client software to read and send email
 - (e.g. Microsoft Outlook, or Netscape Messenger)
 - Simple Mail Transfer Protocol (SMTP)
 - Specifies format of mail messages
 - Post Office Protocol (POP) tells the email server to:
 - Send mail to the user's computer and delete it from the server
 - Send mail to the user's computer and do not delete it from the server
 - Ask whether new mail has arrived.

1.6 Interactive Mail Access Protocol (IMAP)

Newer than POP, provides similar functions with additional features.
 e.g. can send specific messages to the client rather than all the messages.
 A user can view email message headers and the sender's name before
 downloading the entire message.

Allows users to delete and search mailboxes held on the email server.

The disadvantage of POP: You can only access messages from one PC.

The disadvantage of IMAP : Since email is stored on the email server, there is a need for more and more expensive (high speed) storage space.

1.7 World Wide Web: comprises software (Web server and browser) and data (Web sites).

Internet Protocol (IP) Addresses:

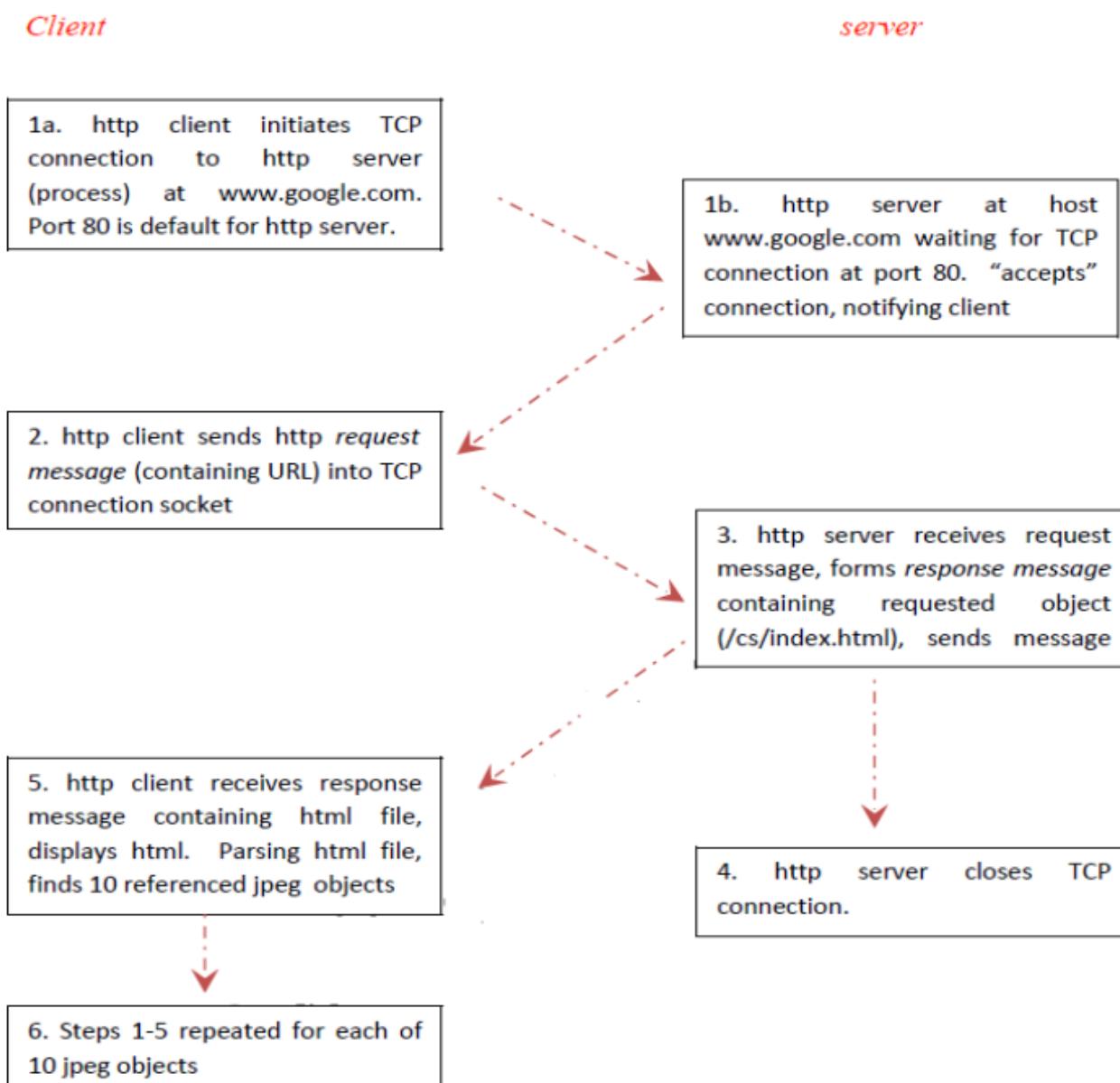
- Every node has a unique numeric address
- Form: 32-bit binary number
- New standard, IPv6, has 128 bits (1998)
- Organizations are assigned groups of IPs for their computers
- **Domain names**
 - Form: host-name. domain-names
 - First domain is the smallest (Google)
 - Last domain specifies the type of organization (.com)
 - Fully qualified domain name - the host name and all of the domain names
 - DNS servers - convert fully qualified domain names to IPs

1.8 HTTP:

- Hypertext Transfer Protocol (HTTP) is the communication protocol used by the Internet to transfer hypertext documents.
- A protocol to transfer hypertext requests and information between servers and browsers

- Hypertext is text, displayed on a computer, with references (hyperlinks) to other text that the reader can immediately follow, usually by a mouse. HTTP is behind every request for a web document or graph, every click of a hypertext link, and every submission of a form.
- HTTP specifies how clients **request** data, and how servers **respond** to these requests.
- The client makes a request for a given page and the server is responsible for finding it and returning it to the client.
- The browser connects and requests a page from the server.
- The server reads the page from the file system and sends it to the client and then terminates the connection

HTTP Transactions

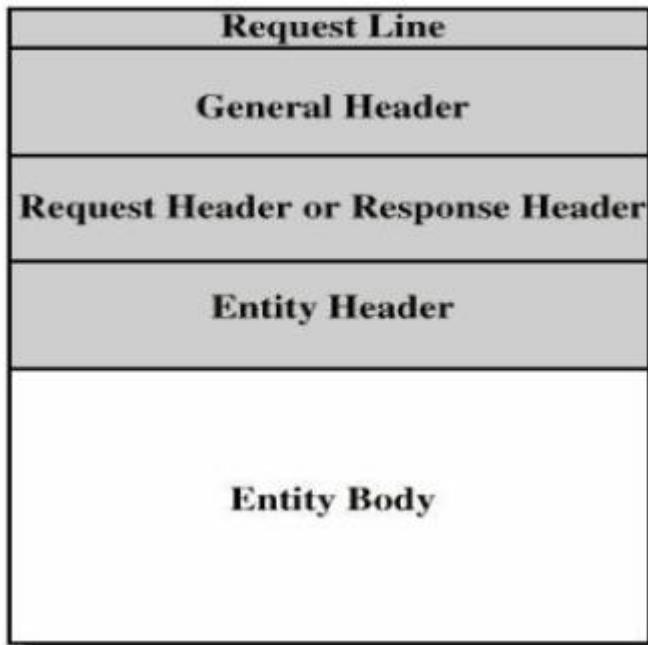


1.9 HTTP Message:

HTTP message is the information transaction between the client and server.

Two types of HTTP Message:

1. Requests
 - a. Client to server
2. Responses
 - a. Server to client



Fields

- Request line or Response line
- General header
- Request header or Response header
- Entity header
- Entity body

.10 Request Message:

Request Line:

- A request line has three parts, separated by spaces
 - o a *method* name
 - o the local path of the requested resource
 - o the version of HTTP being used
- A typical request line is:
 - o GET /path/to/file/index.html HTTP/1.1
- Notes:
 - o **GET** is the most common HTTP method; it says "give me this resource". Other methods include **POST** and **HEAD**. Method names are always uppercase
 - o The path is the part of the URL after the host name, also called the *request URI*
 - o The HTTP version always takes the form "**HTTP/x.x**", uppercase.

Request Header:

HTTP Request Headers

Header	Description
From	Email address of user
User-Agent	Client s/w
Accept File	File types that client will accept
Accept-encoding	Compression methods
Accept-Language	Languages
Referrer	URL of the last document the client displayed
If-Modified-Since	Return document only if modified since specified
Content-length	Length (in bytes) of data to follow

HTTP Request

Method File name HTTP version

```

GET /msaleh/index.html HTTP/1.1
Host: staff.ifm.ac.tz
Connection: close
Accept: text/xml, text/html, text/plain, image/png, */*
Accept-Language: en-GB, en
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Accept-Charset: ISO-8859-1, utf-8; q=0.7, *
If-Modified-Since: Mon, 18 Sep 2006 22:57:19 GMT
Referer: http://web-sniffer.net

```

Blank line

Headers

Data – none for GET

.11 Response Message:

Response Line:

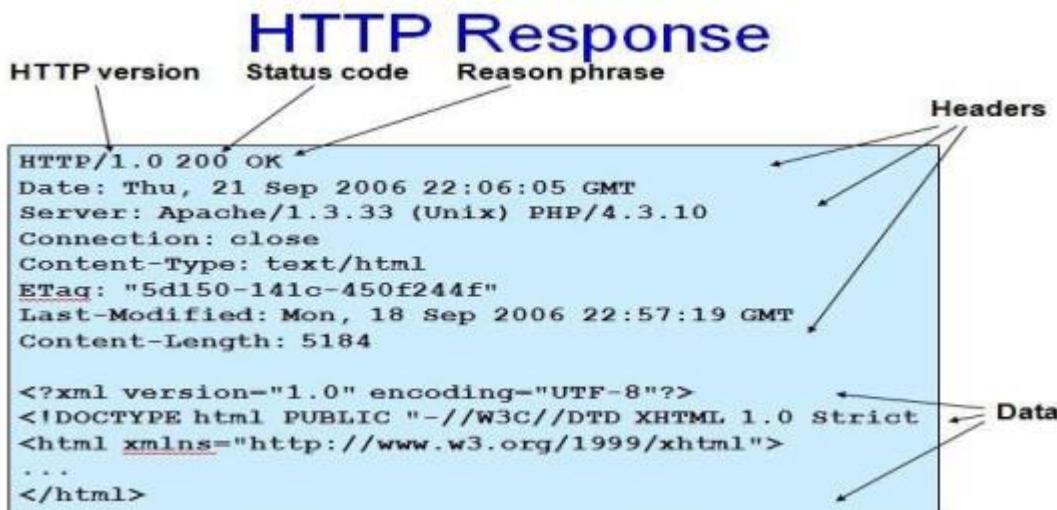
- A request line has three parts, separated by spaces
 - the HTTP version,
 - a *response status code* that gives the result of the request, and
 - an English *reason phrase* describing the status code
- Typical status lines are:
 - HTTP/1.0 200 OK or
 - HTTP/1.0 404 Not Found
- Notes:
 - The HTTP version is in the same format as in the request line, "HTTP/x.x".

- The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary.

HTTP Request Header:

HTTP Response Headers	
Header	Description
Server	Server software
Date	Current Date
Last-Modified	Modification date of document
Expires	Date at which document expires
Location	The location of the document in redirection responses
Pragma	A hint, e.g., no cache
MIME-version	
Link	URL of document's parent
Content-Length	Length in bytes
Allowed	Requests that user can issue, e.g., GET

EXAMPLE



HTTP Method:

- HTTP method is supplied in the request line and specifies the operation that the client has requested.

Some common methods:

- Options
- Get
- Head
- Post
- Put
- Move
- Delete

Two methods that are mostly used are the GET and POST:

- GET for queries that can be safely repeated

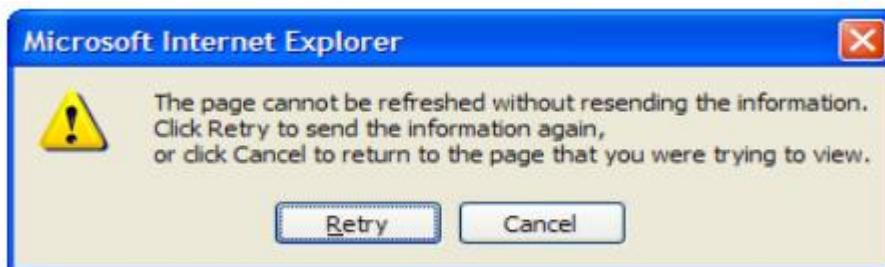
- o **POST** for operations that may have side effects (e.g. ordering a book from an on-line store).

The GET Method

- It is used to retrieve information from a specified URI and is assumed to be a safe, repeatable operation by browsers, caches and other HTTP aware components
- Operations have no side effects and GET requests can be re-issued.
- For example, displaying the balance of a bank account has no effect on the account and can be safely repeated.
- Most browsers will allow a user to refresh a page that resulted from a **GET**, without displaying any kind of warning
- Proxies may automatically retry **GET** requests if they encounter a temporary network connection problem.
- GET requests is that they can only supply data in the form of parameters encoded in the URI (known as a **Query String**) – [downside]
- Cannot be unused for uploading files or other operations that require large amounts of data to be sent to the server.

The POST Method

- Used for operations that have side effects and cannot be safely repeated.
- For example, transferring money from one bank account to another has side effects and should not be repeated without explicit approval by the user.
- If you try to refresh a page in Internet Explorer that resulted from a **POST**, it displays the following message to warn you that there may



The **POST** request message has a content body that is normally used to send parameters and data

- The IIS server returns two status codes in its response for a **POST** request
 - o The first is **100 Continue** to indicate that it has successfully received the **POST** request
 - o The second is **200 OK** after the request has been processed.

HTTP response status codes

- Informational (1xx)
- Successful (2xx)
- Redirection (3xx)
 - o 301: moved permanently
 - Client error (4xx)
 - o 403 : forbidden
 - o 404: Not found
 - Server error (5xx)
 - o 503: Service unavailable
 - o 505: HTTP version not supported

1.12 HTTP

❖ □ Features

- Persistent TCP Connections: Remain open for multiple requests
- Partial Document Transfers: Clients can specify start and stop positions
- Conditional Fetch: Several additional conditions

- Better content negotiation
- More flexible authentication.

❖ □ Web Browsers :

- Mosaic - NCSA (Univ. of Illinois), in early 1993 First to use a GUI, led to Explosion of Web use Initially for X-Windows, under UNIX, but was ported to other platforms by late 1993
- Browsers are clients - always initiate, servers react (although sometimes servers require responses)
- Most requests are for existing documents, using Hypertext Transfer Protocol (HTTP)
- But some requests are for program execution, with the output being returned as a document.

Browser: A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.

❖ □ Web Servers:

- Provide responses to browser requests, either existing documents or dynamically built documents.
- Browser-server connection is now maintained through more than one request- Response cycle
- All communications between browsers and servers use Hypertext Transfer Protocol
- Web servers run as background processes in the operating system.
- Monitor a communications port on the host, accepting HTTP messages when they appear

All current Web servers came from either

1. The original from CERN
 2. The second one, from NCSA
- Web servers have two main directories:
 1. Document root (servable documents)
 2. Server root (server system software)
 - Document root is accessed indirectly by clients
 - Its actual location is set by the server Configuration file
 - Requests are mapped to the actual location
 - Virtual document trees
 - Virtual hosts
 - Proxy servers
 - Web servers now support other Internet protocols
 - Apache (open source, fast, reliable)
 - IIS
 - Maintained through a program with a GUI interface.

HTML 5

HTML is the main markup language for describing the structure of web pages.

HTML stands for HyperText Markup Language. HTML is the basic building block of World Wide Web.

Hypertext is text displayed on a computer or other electronic device with references to other text that the user can immediately access, usually by a mouse click or key press.

Apart from text, hypertext may contain tables, lists, forms, images, and other presentational elements. It is an easy-to-use and flexible format to share information over the Internet.

Markup languages use sets of markup tags to characterize text elements within a document, which gives instructions to the web browsers on how the document should appear.

HTML was originally developed by Tim Berners-Lee in 1990. He is also known as the father of the web. In 1996, the World Wide Web Consortium (W3C) became the authority to maintain the HTML specifications. HTML also became an international standard (ISO) in 2000. HTML5 is the latest version of HTML. HTML5 provides a faster and more robust approach to web development.

HTML Tags and Elements

HTML is written in the form of HTML elements consisting of markup tags. These markup tags are the fundamental characteristic of HTML. Every markup tag is composed of a keyword, surrounded by angle brackets, such as `<html>`, `<head>`, `<body>`, `<title>`, `<p>`, and so on.

HTML tags normally come in pairs like `<html>` and `</html>`. The first tag in a pair is often called the opening tag (or start tag), and the second tag is called the closing tag (or end tag).

An opening tag and a closing tag are identical, except a slash (/) after the opening angle bracket of the closing tag, to tell the browser that the command has been completed.

Inserting Images into Web Pages

Images enhance visual appearance of the web pages by making them more interesting and colorful.

The `` tag is used to insert images in the HTML documents. It is an empty element and contains attributes only. The syntax of the `` tag can be given with:

```

```

The following example inserts three images on the web page:

Example

[Try this code »](#)

```



```

Each image must carry at least two attributes: the `src` attribute, and an `alt` attribute.

The `src` attribute tells the browser where to find the image. Its value is the URL of the image file.

Whereas, the `alt` attribute provides an alternative text for the image, if it is unavailable or cannot be displayed for some reason. Its value should be a meaningful substitute for the image.

HTML Tables

Creating Tables in HTML

HTML table allows you to arrange data into rows and columns. They are commonly used to display tabular data like product listings, customer's details, financial reports, and so on.

You can create a table using the `<table>` element. Inside the `<table>` element, you can use the `<tr>` elements to create rows, and to create columns inside a row you can use the `<td>` elements. You can also define a cell as a header for a group of table cells using the `<th>` element.

The following example demonstrates the most basic structure of a table.

Example

```
<table>
  <tr>
    <th>No.</th>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Peter Parker</td>
    <td>16</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Clark Kent</td>
    <td>34</td>
  </tr>
</table>
```

Tables do not have any borders by default. You can use the CSS `border` property to add borders to the tables. Also, table cells are sized just large enough to fit the contents by default. To add more space around the content in the table cells you can use the CSS `padding` property.

Defining a Table Header, Body, and Footer

HTML provides a series of tags `<thead>`, `<tbody>`, and `<tfoot>` that helps you to create more structured table, by defining header, body and footer regions, respectively.

The following example demonstrates the use of these elements.

Example

```
<table>
  <thead>
    <tr>
      <th>Items</th>
      <th>Expenditure</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Stationary</td>
      <td>2,000</td>
    </tr>
    <tr>
      <td>Furniture</td>
      <td>10,000</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <th>Total</th>
      <td>12,000</td>
    </tr>
  </tfoot>
```

```
</tr>
</tfoot>
</table>
```

HTML Lists

HTML lists are used to present list of information in well formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning.

- **Unordered list** — Used to create a list of related items, in no particular order.
- **Ordered list** — Used to create a list of related items, in a specific order.
- **Description list** — Used to create a list of terms and their descriptions.

HTML Unordered Lists

An unordered list created using the `` element, and each list item starts with the `` element.

The list items in unordered lists are marked with bullets. Here's an example:

Example

```
<ul>
  <li>Chocolate Cake</li>
  <li>Black Forest Cake</li>
  <li>Pineapple Cake</li>
</ul>
```

— The output of the above example will look something like this:

- Chocolate Cake
- Black Forest Cake
- Pineapple Cake

You can also change the bullet type in your unordered list using the CSS `list-style-type` property. The following style rule changes the type of bullet from the default `disc` to `square`:

Example

```
ul {
  list-style-type: square;
}
```

Please check out the tutorial on [CSS lists](#) to learn about styling HTML lists in details.

HTML Ordered Lists

An ordered list created using the `` element, and each list item starts with the `` element. Ordered lists are used when the order of the list's items is important.

The list items in an ordered list are marked with numbers. Here's an example:

Example

```
<ol>
  <li>Fasten your seatbelt</li>
  <li>Starts the car's engine</li>
  <li>Look around and go</li>
</ol>
```

— The output of the above example will look something like this:

1. Fasten your seatbelt
2. Starts the car's engine
3. Look around and go

HTML5 Image

HTML Images Syntax

In HTML, images are defined with the `` tag.

The `` tag is empty, it contains attributes only, and does not have a closing tag.

The `src` attribute specifies the URL (web address) of the image:

```

```

EXAMPLE

```
<!DOCTYPE html>

<html>

<body>

<h2>HTML Image</h2>



</body>

</body>

</html>
```

OUTPUT

HTML Image



HTML Form

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called **controls** like *inputbox*, *checkboxes*, *radio-buttons*, *submit buttons*, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

The `<form>` tag is used to create an HTML form. Here's a simple example of a login form:

Example

```
<form>
  <label>Username: <input type="text"></label>
  <label>Password: <input type="password"></label>
  <input type="submit" value="Submit">
</form>
```

The following section describes different types of controls that you can use in your form.

Input Element

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the `type` attribute. An input element can be of type *text field*, *password field*, *checkbox*, *radio button*, *submit button*, *reset button*, *file select box*, as well as several [new input types](#) introduced in HTML5.

The most frequently used input types are described below.

Text Fields

Text fields are one line areas that allow the user to input text.

Single-line text input controls are created using an `<input>` element, whose `type` attribute has a value of `text`. Here's an example of a single-line text input used to take username:

Example

```
<form>
  <label for="username">Username:</label>
  <input type="text" name="username" id="username">
</form>
```

— The output of the above example will look something like this:

Username:

Password Field

Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots. This is to prevent someone else from reading the password on the screen. This is also a single-line text input controls created using an `<input>` element whose `type` attribute has a value of `password`.

Example

```
<form>
  <label for="user-pwd">Password:</label>
  <input type="password" name="user-password" id="user-pwd">
</form>
```

— The output of the above example will look something like this:

Password:

Radio Buttons

Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `radio`.

Example

[Try this code »](#)

```
<form>
  <input type="radio" name="gender" id="male">
  <label for="male">Male</label>
  <input type="radio" name="gender" id="female">
  <label for="female">Female</label>
</form>
```

— The output of the above example will look something like this:

Male Female

Checkboxes

Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `checkbox`.

Example

```
<form>
  <input type="checkbox" name="sports" id="soccer">
  <label for="soccer">Soccer</label>
  <input type="checkbox" name="sports" id="cricket">
```

```
<label for="cricket">Cricket</label>
<input type="checkbox" name="sports" id="baseball">
<label for="baseball">Baseball</label>
</form>
```

— The output of the above example will look something like this:

Soccer Cricket Baseball

File Select box

The file fields allow a user to browse for a local file and send it as an attachment with the form data. Web browsers such as Google Chrome and Firefox render a file select input field with a Browse button that enables the user to navigate the local hard drive and select a file.

This is also created using an `<input>` element, whose type attribute value is set to file.

Example

```
<form>
  <label for="file-select">Upload:</label>
  <input type="file" name="upload" id="file-select">
</form>
```

— The output of the above example will look something like this:

Upload: No file chosen

Textarea

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an `<textarea>` element.

Example

```
<form>
  <label for="address">Address:</label>
  <textarea rows="3" cols="30" name="address" id="address"></textarea>
</form>
```

— The output of the above example will look something like this:

Address: 

Select Boxes

A select box is a dropdown list of options that allows user to select one or more option from a pull-down list of options. Select box is created using the `<select>` element and `<option>` element.

The `<option>` elements within the `<select>` element define each list item.

Example

```
<form>
  <label for="city">City:</label>
  <select name="city" id="city">
    <option value="sydney">Sydney</option>
    <option value="melbourne">Melbourne</option>
    <option value="cromwell">Cromwell</option>
  </select>
</form>
```

— The output of the above example will look something like this:

City:

Submit and Reset Buttons

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's `action` attribute to process the submitted data.

A reset button resets all the forms control to default values. Try out the following example by typing your name in the text field, and click on submit button to see it in action.

Example

```
<form action="action.php" method="post">
  <label for="first-name">First Name:</label>
  <input type="text" name="first-name" id="first-name">
  <input type="submit" value="Submit">
  <input type="reset" value="Reset">
</form>
```

First Name:

HTML5 Colors

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

<h1 style="background-color:Tomato;">Tomato</h1>
<h1 style="background-color:Orange;">Orange</h1>
<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>
<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>
```

```
<h1 style="background-color:Gray;">Gray</h1>  
<h1 style="background-color:SlateBlue;">SlateBlue</h1>  
<h1 style="background-color:Violet;">Violet</h1>  
<h1 style="background-color:LightGray;">LightGray</h1>  
  
</body>  
</html>
```

OUTPUT



The image shows a vertical stack of colored bars, each representing the background color of an H1 tag from the provided code. From top to bottom, the colors are: Tomato (red), Orange (yellow), DodgerBlue (blue), MediumSeaGreen (green), Gray (gray), SlateBlue (purple), Violet (pink), and LightGray (light gray). The bars are separated by thin white lines.

Tomato
Orange
DodgerBlue
MediumSeaGreen
Gray
SlateBlue
Violet
LightGray

HTML5 Audio

Embedding Audio in HTML Document

Inserting audio onto a web page was not easy before, because web browsers did not have a uniform standard for defining embedded media files like audio.

Using the HTML5 audio Element

The newly introduced HTML5 `<audio>` element provides a standard way to embed audio in web pages. However, the audio element is relatively new but it works in most of the modern web browsers.

The following example simply inserts an audio into the HTML5 document, using the browser default set of controls, with one source defined by the `src` attribute.

Example

```
<audio controls="controls" src="media/birds.mp3">
  Your browser does not support the HTML5 Audio element.
</audio>
```

An audio, using the browser default set of controls, with alternative sources.

Example

```
<audio controls="controls">
  <source src="media/birds.mp3" type="audio/mpeg">
  <source src="media/birds.ogg" type="audio/ogg">
  Your browser does not support the HTML5 Audio element.
</audio>
```

HTML5 Video

Embedding Video in HTML Document

Inserting video onto a web page was not relatively easy, because web browsers did not have a uniform standard for defining embedded media files like video.

Using the HTML5 video Element

The newly introduced HTML5 `<video>` element provides a standard way to embed video in web pages. However, the video element is relatively new, but it works in most of the modern web browsers.

The following example simply inserts a video into the HTML document, using the browser default set of controls, with one source defined by the `src` attribute.

Example

```
<video controls="controls" src="media/shuttle.mp4">
  Your browser does not support the HTML5 Video element.
</video>
```

A video, using the browser default set of controls, with alternative sources.

Example

```
<video controls="controls">
  <source src="media/shuttle.mp4" type="video/mp4">
  <source src="media/shuttle.ogv" type="video/ogg">
  Your browser does not support the HTML5 Video element.
</video>
```

New HTML5 Elements

The most interesting new HTML5 elements are:

New **semantic elements** like `<header>`, `<footer>`, `<article>`, and `<section>`.

New **attributes of form elements** like number, date, time, calendar, and range.

New **graphic elements**: `<svg>` and `<canvas>`.

New **multimedia elements**: `<audio>` and `<video>`.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: `<div>` and `` - Tells nothing about its content.

Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

New Semantic Elements in HTML5

Many web sites contain HTML code like:

```
<div id="nav"><div class="header"><div id="footer">  
to indicate navigation, header, and footer.
```

HTML5 offers new semantic elements to define different parts of a web page:

- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`
- `<figure>`
- `<footer>`
- `<header>`
- `<main>`
- `<mark>`
- `<nav>`
- `<section>`
- `<summary>`
- `<time>`



HTML5 <section> Element

The `<section>` element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

A home page could normally be split into sections for introduction, content, and contact information.

Example

```

<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>

```

HTML5 <article> Element

The `<article>` element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an `<article>` element can be used:

- Forum post
- Blog post
- Newspaper article

Example

```

<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>

```

HTML5 <header> Element

The `<header>` element specifies a header for a document or section.

The `<header>` element should be used as a container for introductory content.

You can have several `<header>` elements in one document.

The following example defines a header for an article:

Example

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML5 `<footer>` Element

The `<footer>` element specifies a footer for a document or section.

A `<footer>` element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You may have several `<footer>` elements in one document.

Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

HTML5 `<figure>` and `<figcaption>` Elements

The purpose of a figure caption is to add a visual explanation to an image.

In HTML5, an image and a caption can be grouped together in a `<figure>` element:

Example

```
<figure>
  
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

OUTPUT

Places to Visit

Puglia's most famous sight is the unique conical houses (Trulli) found in the area around Alberobello, a declared UNESCO World Heritage Site.



Fig.1 - Trulli, Puglia, Italy.

Semantic Elements in HTML5

Below is an alphabetical list of the new semantic elements in HTML5.

The links go to our complete [HTML5 Reference](#).

Tag	Description
<code><article></code>	Defines an article
<code><aside></code>	Defines content aside from the page content
<code><details></code>	Defines additional details that the user can view or hide
<code><figcaption></code>	Defines a caption for a <code><figure></code> element

<u><figure></u>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<u><footer></u>	Defines a footer for a document or section
<u><header></u>	Specifies a header for a document or section
<u><main></u>	Specifies the main content of a document
<u><mark></u>	Defines marked/highlighted text
<u><nav></u>	Defines navigation links
<u><section></u>	Defines a section in a document
<u><summary></u>	Defines a visible heading for a <details> element
<u><time></u>	Defines a date/time

HTML5 Drag and Drop



Drag the W3Schools image into the rectangle.

Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

In HTML5, drag and drop is part of the standard: Any element can be draggable.

HTML Drag and Drop Example

The example below is a simple drag and drop example:

Example

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>



</body>
</html>
```

OUTPUT

Drag the W3Schools image into the rectangle:



HTML5 <nav> Element

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a `<nav>` element. The `<nav>` element is intended only for major block of navigation links.

Example

```
<nav>
<a href="/html/">HTML</a> |
<a href="/css/">CSS</a> |
<a href="/js/">JavaScript</a> |
<a href="/jquery/">jQuery</a>
</nav>
```

What Is CSS3 And Why Is It Used?

To help build highly interactive online pages, CSS3 is invariably used due to its importance in providing greater options in the design process. When marketing products and services, web design plays a vital part; a site should be created in a manner that will draw potential customers to explore and revisit a website more often. Many **web design firms** are developing and enhancing websites through the use of CSS3 as this is a great form of web development. This article will help define CSS3 and will point out its advantages.

Definition

The acronym CSS stands for Cascading Style Sheets which is used to augment the functionality, versatility, and efficient performance of site content. It allows for the creation of content-rich websites that do not require much weight or codes; this translates into more interactive graphics and animation, superior user-interface, and significantly more organization and rapid download time.

It is used with HTML to create content structure, with CSS3 being used to format structured content. It is responsible for font properties, colors, text alignments, graphics, background images, tables and other components. This tool provides extra capabilities such as absolute, fixed and relative positioning of various elements. The increasing popularity of CSS3 when used by **web design firms** stimulates major browsers such as Google Chrome, Firefox, Safari, and IE9 to adopt and embrace this programming language.

Advantages

Although CSS3 is not the only web development solution, it does allow provide greater advantages for several reasons.

- **Customization** – A web page can be customized and alterations created in the design by simply changing a modular file.
- **Bandwidth Requirements** – It decreases server bandwidth requirements, giving rapid download time when a site is accessed with desktop or hand-held devices, providing an improved user experience.
- **Consistency** – It delivers consistent and accurate positioning of navigational elements on the website.
- **Appealing** – It makes the site more appealing with adding videos and graphics easier.
- **Viewing** – It allows online videos to be viewed without the use of third-party plug-ins.
- **Visibility** – It delivers the opportunity to improve brand visibility by designing effective online pages.
- **Cost Effective** – It is cost-effective, time-saving, and supported by most browsers.

Since the introduction of CSS3, there is greater control of the presentation of content and various elements on a website; however it is not really responsible for overall design as it only specifies the structure and content presentation of certain web pages.

External, internal, and inline CSS styles

Cascading Style Sheets (CSS) are files with styling rules that govern how your website is presented on screen. CSS rules can be applied to your website's HTML files in various ways. You can use an **external stylesheet**, an **internal stylesheet**, or an **inline style**. Each method has advantages that suit particular uses.

An **external stylesheet** is a standalone .css file that is linked from a web page. The advantage of external stylesheets is that it can be created once and the rules applied to multiple web pages. Should you need to make widespread changes to your site design, you can make a single change in the stylesheet and it will be applied to all linked pages, saving time and effort.

An **internal stylesheet** holds CSS rules for the page in the **head** section of the HTML file. The rules only apply to that page, but you can configure CSS classes and IDs that can be used to style multiple elements in the page code. Again, a single change to the CSS rule will apply to all tagged elements on the page.

Inline styles relate to a specific HTML tag, using a **style** attribute with a CSS rule to style a specific page element. They're useful for quick, permanent changes, but are less flexible than external and internal stylesheets as each inline style you create must be separately edited should you decide to make a design change.

Using external CSS stylesheets

An HTML page styled by an external CSS stylesheet must reference the .css file in the document head. Once created, the CSS file must be uploaded to your server and linked in the HTML file with code such as:

```
<link href="style.css" rel="stylesheet" type="text/css">
```

You can name your stylesheet whatever you wish, but it should have a .css file extension.

Using internal CSS stylesheets

Rather than linking an external .css file, HTML files using an internal stylesheet include a set of rules in their **head** section. CSS rules are wrapped in **<style>** tags, like this:

```
<head>
<style type="text/css">

h1 {
    color:#fff
    margin-left: 20px;
}

p {
```

```
font-family: Arial, Helvetica, Sans Serif;  
}  
  
</style>  
</head>
```

Using inline styles

Inline styles are applied directly to an element in your HTML code. They use the **style** attribute, followed by regular CSS properties.

For example:

```
<h1 style="color:red;margin-left:20px;">Today's Update</h1>
```

Rule Cascading

Cascade and inheritance

Conflicting rules

CSS stands for **Cascading Style Sheets**, and that first word *cascading* is incredibly important to understand — the way that the cascade behaves is key to understanding CSS.

At some point, we will find that the CSS have created two rules which could potentially apply to the same element. The **cascade**, and the closely-related concept of **specificity**, are mechanisms that control which rule applies when there is such a conflict. Which rule is styling your element may not be the one you expect, so you need to understand how these mechanisms work.

Also significant here is the concept of **inheritance**, which means that some CSS properties by default inherit values set on the current element's parent element, and some don't. This can also cause some behavior that you might not expect.

The cascade

Stylesheets **cascade** — at a very simple level this means that the order of CSS rules matter; when two rules apply that have equal specificity the one that comes last in the CSS is the one that will be used.

EXAMPLE

In the below example, we have two rules that could apply to the h1. The h1 ends up being colored blue — these rules have an identical selector and therefore carry the same specificity, so the last one in the source order wins.

```
h1 {  
  color: red;  
}
```

```
}
```

```
h1 {
```

```
    color: blue;
```

```
}
```

```
<h1>This is my heading.</h1>
```

OUTPUT

```
This is my heading.
```

Specificity

Specificity is how the browser decides which rule applies if multiple rules have different selectors, but could still apply to the same element. It is basically a measure of how specific a selector's selection will be:

- An element selector is less specific — it will select all elements of that type that appear on a page — so will get a lower score.
- A class selector is more specific — it will select only the elements on a page that have a specific class attribute value — so will get a higher score.

Example time! Below we again have two rules that could apply to the h1. The below h1 ends up being colored red — the class selector gives its rule a higher specificity, and so it will be applied even though the rule with the element selector appears further down in the source order.

EXAMPLE

```
main-heading {
```

```
    color: red;
```

```
}
```

```
h1 {
```

```
    color: blue;
```

```
}
```

```
. <h1 class="main-heading">This is my heading.</h1>
```

OUTPUT

```
This is my heading.
```

Inheritance

Inheritance also needs to be understood in this context — some CSS property values set on parent elements are inherited by their child elements, and some aren't.

For example, if you set a color and font-family on an element, every element inside it will also be styled with that color and font, unless you've applied different color and font values directly to them.

Some properties do not inherit — for example if you set a `width` of 50% on an element, all of its descendants do not get a width of 50% of their parent's width. If this was the case, CSS would be very frustrating to use!

```
body {  
  color: blue;  
}  
  
span {  
  color: black;  
}
```

```
<p>As the body has been set to have a color of blue this is inherited through the  
descendants.</p>
```

```
<p>We can change the color by targetting the element with a selector,  
such as this
```

```
<span>span</span>.</p>
```

OUTPUT

```
As the body has been set to have a color of blue this is inherited through  
the descendants.
```

```
We can change the color by targetting the element with a selector, such as  
this span.
```

CSS3 Shadow Effects

With CSS3 you can create two types of shadows: `text-shadow` (adds shadow to text) and `box-shadow` (adds shadow to other elements).

CSS3 Text Shadow

The `text-shadow` property can take up to four values:

- the horizontal shadow
- the vertical shadow
- the blur effect
- the color

Examples:

- Normal text shadow

```
• h1 {  
  • text-shadow: 2px 2px 5px crimson;
```

```
}
```

CSS3 Text Shadow Effect

- Glowing text effect

```
• h1 {  
  • text-shadow: 0 0 4px #00FF9C;
```

```
}
```

This Title Glows!

CSS3 Box Shadow

The `box-shadow` property can take up to six values:

- (optional) the `inset` keyword (changes the shadow to one inside the frame)
- the horizontal shadow
- the vertical shadow
- the blur effect
- the spreading
- the color

Examples:

```
.first-div {
    box-shadow: 1px 1px 5px 3px grey;
}
```

This is a <div> with a box-shadow.

CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

What are CSS Animations?

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

When you specify CSS styles inside the `@keyframes` rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

Example

```
/* The animation code */
@keyframes example {
```

```

from {background-color: red;}
to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}

```

Note: The `animation-duration` property defines how long time an animation should take to complete. If the `animation-duration` property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

Example

```

/* The animation code */
@keyframes example {
  0% {background-color: red;}
  25% {background-color: yellow;}
  50% {background-color: blue;}
  100% {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}

```

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	Description
@keyframes	Specifies the animation code
animation	A shorthand property for setting all the animation properties

<u>animation-delay</u>	Specifies a delay for the start of an animation
<u>animation-direction</u>	Specifies whether an animation should be played forwards, backwards or in alternate cycles
<u>animation-duration</u>	Specifies how long time an animation should take to complete one cycle
<u>animation-fill-mode</u>	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
<u>animation-iteration-count</u>	Specifies the number of times an animation should be played
<u>animation-name</u>	Specifies the name of the @keyframes animation
<u>animation-play-state</u>	Specifies whether the animation is running or paused
<u>animation-timing-function</u>	Specifies the speed curve of the animation

CSS Transitions

CSS Transitions is a module of CSS that lets you create gradual transitions between the values of specific CSS properties. The behavior of these transitions can be controlled by specifying their timing function, duration, and other attributes.

Properties

- [transition](#)
- [transition-delay](#)
- [transition-duration](#)
- [transition-property](#)
- [transition-timing-function](#)

The **transition** [CSS](#) property is a [shorthand property](#) for [transition-property](#), [transition-duration](#), [transition-timing-function](#), and [transition-delay](#).

CSS transition Property

Example

Hover over a <div> element to gradually change the width from 100px to 300px:

```
div {
  width: 100px;
  transition: width 2s;
}

div:hover {
  width: 300px;
}
```

OUTPUT

The transition Property

Hover over the div element below, to see the transition effect:



Definition and Usage

The **transition** property is a shorthand property for:

- [transition-property](#)
- [transition-duration](#)
- [transition-timing-function](#)

Property Values

Value	Description
transition-property	Specifies the name of the CSS property the transition effect is for
transition-duration	Specifies how many seconds or milliseconds the transition effect takes to complete
transition-timing-function	Specifies the speed curve of the transition effect
transition-delay	Defines when the transition effect will start
initial	Sets this property to its default value. Read about initial
inherit	Inherits this property from its parent element. Read about inherit

Example

When an <input type="text"> gets focus, gradually change the width from 100px to 250px:

```
input[type=text] {  
    width: 100px;  
    transition: width .35s ease-in-out;  
}  
  
input[type=text]:focus {
```

```
width: 250px;  
}
```

OUTPUT

The width Property

Set the width of the input field to 100 pixels. However, when the input field gets focus, make it 250 pixels wide:

Search:

CSS background-color

The `background-color` property specifies the background color of an element.

Example

The background color of a page is set like this:

```
body {  
    background-color: lightblue;  
}
```

CSS background-image

The `background-image` property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

Example

The background image for a page can be set like this:

```
body {  
    background-image: url("paper.gif");  
}
```

CSS background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

The shorthand property for background is `background`.

Example

Use the shorthand property to set all the background properties in one declaration:

```
body {  
    background: #ffffff url("img_tree.png") no-repeat right top;  
}
```

CSS Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The `border` property is a shorthand property for the following individual border properties:

- `border-width`
- `border-style` (required)
- `border-color`

Example

```
p {  
    border: 5px solid red;  
}
```

Result:

Some text

UNIT II - CLIENT SIDE PROGRAMMING

Java Script: An introduction to JavaScript – JavaScript DOM Model – Date and Objects – Regular Expressions – Exception Handling – Validation – Built-in objects – Event Handling – DHTML with JavaScript – JSON introduction – Syntax – Function Files – Http Request – SQL.

PART - A

Q.No	Questions																						
1.	Evaluate various Java Script Object models.																						
2.	Define DOM.																						
3.	<p>Give any four methods of Date objects.</p> <p><u>JavaScript Get Date Methods</u></p> <p>These methods can be used for getting information from a date object:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 5px;">Method</th><th style="text-align: left; padding: 5px;">Description</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;">getFullYear()</td><td style="padding: 5px;">Get the year as a four digit number (yyyy)</td></tr> <tr> <td style="padding: 5px;">getMonth()</td><td style="padding: 5px;">Get the month as a number (0-11)</td></tr> <tr> <td style="padding: 5px;">getDate()</td><td style="padding: 5px;">Get the day as a number (1-31)</td></tr> <tr> <td style="padding: 5px;">getHours()</td><td style="padding: 5px;">Get the hour (0-23)</td></tr> <tr> <td style="padding: 5px;">getMinutes()</td><td style="padding: 5px;">Get the minute (0-59)</td></tr> <tr> <td style="padding: 5px;">getSeconds()</td><td style="padding: 5px;">Get the second (0-59)</td></tr> <tr> <td style="padding: 5px;">getMilliseconds()</td><td style="padding: 5px;">Get the millisecond (0-999)</td></tr> <tr> <td style="padding: 5px;">getTime()</td><td style="padding: 5px;">Get the time (milliseconds since January 1, 1970)</td></tr> <tr> <td style="padding: 5px;">getDay()</td><td style="padding: 5px;">Get the weekday as a number (0-6)</td></tr> <tr> <td style="padding: 5px;">Date.now()</td><td style="padding: 5px;">Get the time. ECMAScript 5.</td></tr> </tbody> </table> <p><u>JavaScript Set Date Methods</u></p> <p>Set Date methods let you set date values (years, months, days, hours, minutes, seconds, milliseconds) for a Date Object.</p>	Method	Description	getFullYear()	Get the year as a four digit number (yyyy)	getMonth()	Get the month as a number (0-11)	getDate()	Get the day as a number (1-31)	getHours()	Get the hour (0-23)	getMinutes()	Get the minute (0-59)	getSeconds()	Get the second (0-59)	getMilliseconds()	Get the millisecond (0-999)	getTime()	Get the time (milliseconds since January 1, 1970)	getDay()	Get the weekday as a number (0-6)	Date.now()	Get the time. ECMAScript 5.
Method	Description																						
getFullYear()	Get the year as a four digit number (yyyy)																						
getMonth()	Get the month as a number (0-11)																						
getDate()	Get the day as a number (1-31)																						
getHours()	Get the hour (0-23)																						
getMinutes()	Get the minute (0-59)																						
getSeconds()	Get the second (0-59)																						
getMilliseconds()	Get the millisecond (0-999)																						
getTime()	Get the time (milliseconds since January 1, 1970)																						
getDay()	Get the weekday as a number (0-6)																						
Date.now()	Get the time. ECMAScript 5.																						

Set Date Methods

Set Date methods are used for setting a part of a date:

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

	<p>Write the JavaScript methods to retrieve the data and time based on the computer locale.</p> <p><script></p> <pre>// Use of Date.now() function var d = Date(Date.now());</pre> <p>4. // Converting the number of millisecond in date string a = d.toString()</p> <pre>// Printing the current date document.write("The current date is: " + a)</pre> <p></script></p> <p>OUTPUT</p> <p>The current date is: Thu Jan 09 2020 20:35:39 GMT+0530 (India Standard Time)</p>
5.	Can you list the different methods defined in document and window object of JavaScript.

Window Object

The window object represents an open window in a browser.

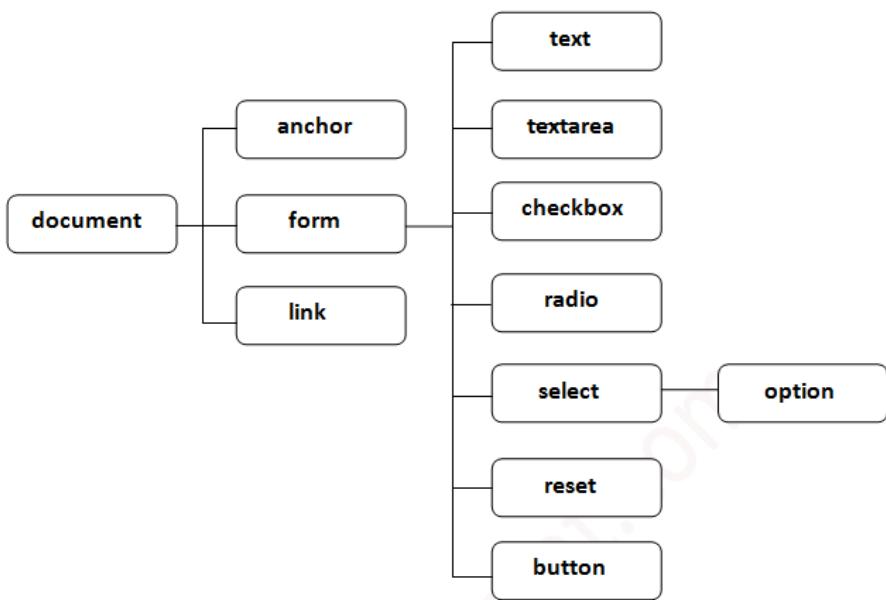
Window Object Methods

Method	Description
alert()	Displays an alert box with a message and an OK button
atob()	Decodes a base-64 encoded string
blur()	Removes focus from the current window
btoa()	Encodes a string in base-64
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
focus()	Sets focus to the current window
getComputedStyle()	Gets the current computed CSS styles applied to an element
getSelection()	Returns a Selection object representing the range of text selected by the user
matchMedia()	Returns a MediaQueryList object representing the specified CSS media query string
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input

	<code>requestAnimationFrame()</code>	Requests the browser to call a function to update an animation before the next repaint
	<code>resizeBy()</code>	Resizes the window by the specified pixels
	<code>resizeTo()</code>	Resizes the window to the specified width and height
	<code>scroll()</code>	Deprecated. This method has been replaced by the <code>scrollTo()</code> method.
	<code>scrollBy()</code>	Scrolls the document by the specified number of pixels
	<code>scrollTo()</code>	Scrolls the document to the specified coordinates
	<code>setInterval()</code>	Calls a function or evaluates an expression at specified intervals (in milliseconds)
	<code>setTimeout()</code>	Calls a function or evaluates an expression after a specified number of milliseconds
	<code>stop()</code>	Stops the window from loading

Document Object Model

1. [Document Object](#)
 2. [Properties of document object](#)
 3. [Methods of document object](#)
 4. [Example of document object](#)
- The **document object** represents the whole html document.
 - When html document is loaded in the browser, it becomes a document object.
 - It is the **root element** that represents the html document. It has properties and methods.
 - By the help of document object, we can add dynamic content to our web page
 - According to W3C - "*The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.*"



Properties of document object

Let's see the properties of document object that can be accessed and modified by the document object.

Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the document.
writeln("string")	writes the given string on the document with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

6.	<p>Name which parser is best in parsing in large size documents. Why?</p>
7.	<p>Summarize benefits of using JavaScript code in an HTML document.</p> <h2>Advantages of JavaScript</h2> <p>The merits of using JavaScript are –</p> <ul style="list-style-type: none"> • Less server interaction – You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server. • Immediate feedback to the visitors – They don't have to wait for a page reload to see if they have forgotten to enter something. • Increased interactivity – You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard. • Richer interfaces – You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.
8.	<p>Predict the need for client and server side scripting.</p> <p>Client-side scripting (embedded scripts) is code that exists inside the client's HTML page. This code will be processed on the client machine and the HTML page will NOT perform a PostBack to the web-server. Traditionally, client-side scripting is used for page navigation, data validation and formatting. The language used in this scripting is JavaScript. JavaScript is compatible and is able to run on any internet browser.</p> <p>The two main benefits of client-side scripting are:</p> <ol style="list-style-type: none"> 1. The user's actions will result in an immediate response because they don't require a trip to the server. 2. Fewer resources are used and needed on the web-server. <p>Server-side scripting is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user's (client's) request to the website. The alternative is for the web server itself to deliver a static web page.</p> <p>The client-side script executes the code to the client side which is visible to the users while a server-side script is executed in the server end which users cannot see.</p>
9.	<p>Interpret how exceptions are handled in Java script.</p>

The try...catch...finally Statement

The latest versions of JavaScript added exception handling capabilities. JavaScript implements the **try...catch...finally** construct as well as the **throw** operator to handle exceptions.

You can **catch** programmer-generated and **runtime** exceptions, but you cannot **catch** JavaScript syntax errors.

Here is the **try...catch...finally** block syntax –

```
<script type = "text/javascript">
<!--
try {
    // Code to run
    [break;]
}

catch ( e ) {
    // Code to run if an exception occurs
    [break;]
}

[ finally {
    // Code that is always executed regardless of
    // an exception occurring
}]
//-->
</script>
```

The **try** block must be followed by either exactly one **catch** block or one **finally** block (or one of both). When an exception occurs in the **try** block, the exception is placed in **e** and the

catch block is executed. The optional **finally** block executes unconditionally after try/catch.

Examples

Here is an example where we are trying to call a non-existing function which in turn is raising an exception. Let us see how it behaves without **try...catch** –

```
<html>
<head>
<script type = "text/javascript">
<!--
function myFunc() {
    var a = 100;
    alert("Value of variable a is : " + a );
}
//-->
```

```

</script>
</head>

<body>
<p>Click the following to see the result:</p>

<form>
<input type = "button" value = "Click Me" onclick = "myFunc();"/>
</form>
</body>
</html>

```

Define JavaScript statement with an example.

JavaScript Statements

Example

```

var x, y, z;      // Statement 1
x = 5;            // Statement 2
y = 6;            // Statement 3
z = x + y;        // Statement 4

```

JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

10.

```

<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Statements</h2>

```

<p>In HTML, JavaScript statements are executed by the browser.</p>

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

<script>

```
document.getElementById("demo").innerHTML = "Hello Dolly.;"
```

</script>

</body>

</html>

OUTPUT

	<p>JavaScript Statements</p> <p>In HTML, JavaScript statements are executed by the browser.</p> <p>Hello Dolly.</p>
	<p>Point out any two techniques of event programming.</p> <h2>What is an Event ?</h2> <p>JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.</p> <p>When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.</p> <p>Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.</p> <p>Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.</p> <p>Please go through this small tutorial for a better understanding HTML Event Reference. Here we will see a few examples to understand a relation between Event and JavaScript –</p> <h2>onclick Event Type</h2>
11.	<p>This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.</p> <p>Example</p> <p>Try the following example.</p> <pre><html> <head> <script type = "text/javascript"> <!-- function sayHello() { alert("Hello World") } //--> </script> </head></pre>

```

<body>
    <p>Click the following button and see result</p>
    <form>
        <input type = "button" onclick = "sayHello()" value = "Say Hello" />
    </form>
</body>
</html>

```

Example 2

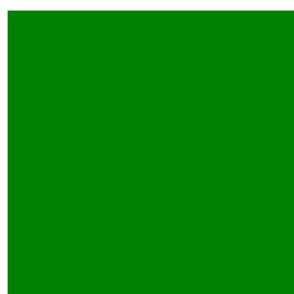
```

<!doctype html>
<html>
<head>
    <script>
        function hov() {
            var e = document.getElementById('hover');
            e.style.display = 'none';
        }
    </script>
</head>
<body>
    <div id="hover" onmouseover="hov()"
        style="background-color:green;height:200px;width:200px;">
    </div>
</body>
</html>

```

OUTPUT

Before mouse is taken over green square-



Green square gets disappear after mouse is taken over it.

What is DHTML?

DHTML

12.

DHTML stands for Dynamic HTML, it is totally different from HTML. The DHTML make use of Dynamic object model to make changes in settings and also in properties and methods.

	<p>DHTML allows different scripting languages in a web page to change their variables, which enhance the effects, looks and many others functions after the whole page have been fully loaded or under a view process, or otherwise static HTML pages on the same.</p> <p>DHTML is used to create interactive and animated web pages that are generated in real-time, also known as dynamic web pages so that when such a page is accessed, the code within the page is analyzed on the web server and the resulting HTML is sent to the client's web browser.</p>
13.	<p>Differentiate HTML and DHTML</p> <p>Some differences between HTML and DHTML:</p> <ul style="list-style-type: none"> • HTML is a mark-up language, while DHTML is a collection of technology. • DHTML creates dynamic web pages, whereas HTML creates static web pages. • DHTML allows including small animations and dynamic menus in Web pages. • DHML used events, methods, properties to insulate dynamism in HTML Pages. • DHML is basically using JavaScript and style sheets in an HTML page. • HTML sites will be slow upon client-side technologies, while DHTML sites will be fast enough upon client-side technologies. • HTML creates a plain page without any styles and Scripts called as HTML. Whereas, DHTML creates a page with HTML, CSS, DOM and Scripts called as DHTML. • HTML cannot have any server side code but DHTML may contain server side code. • In HTML, there is no need for database connectivity, but DHTML may require connecting to a database as it interacts with user. • HTML files are stored with .htm or .html extension, while DHTML files are stored with .dhtm extension. • HTML does not require any processing from browser, while DHTML requires processing from browser which changes its look and feel.
14.	<p>Point out the key features of DHTML.</p> <p>Key Features: Following are the some major key features of DHTML:</p> <ul style="list-style-type: none"> • Tags and their properties can be changed using DHTML. • It is used for real-time positioning. • Dynamic fonts can be generated using DHTML. • It is also used for data binding. • It makes a webpage dynamic and be used to create animations, games, applications along with providing new ways of navigating through websites. • The functionality of a webpage is enhanced due to the usage of low-bandwidth effect by DHTML. • DHTML also facilitates the use of methods, events, properties, and codes.
15.	<p>Classify the data types in JSON</p> <p>JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. ... JSON is built on two structures: A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.</p>

	<p>JSON Data Types. At the granular level, JSON consist of 6 data types. First four data types (string, number, boolean and null) can be referred as simple data types. Other two data types (object and array) can be referred as complex data types.</p>
	<p>How to convert text into a JavaScript object using JSON?</p> <h2><u>Converting JSON text to JavaScript Object</u></h2> <p>JSON (JavaScript Object Notation) is a lightweight data-interchange format. As its name suggests, JSON is derived from the JavaScript programming language, but it's available for use by many languages including Python, Ruby, PHP, and Java and hence, it can be said as language-independent. For humans, it is easy to read and write and for machines, it is easy to parse and generate. It is very useful for storing and exchanging data.</p> <p>A JSON object is a key-value data format that is typically rendered in curly braces. JSON object consist of curly braces ({ }) at the either ends and have key-value pairs inside the braces. Each key-value pair inside braces are separated by comma (,). JSON object looks something like this :</p> <pre>{ "key": "value", "key": "value", "key": "value", }</pre> <p>16. Example for a JSON object :</p> <pre>{ "rollno": 101, "name": "Mayank", "age": 20, }</pre> <h3>Conversion of JSON text to Javascript Object</h3> <p>JSON text/object can be converted into Javascript object using the function JSON.parse().</p> <pre>var object1 = JSON.parse('{"rollno":101, "name":"Mayank", "age":20}');</pre> <p>For getting the value of any key from a Javascript object, we can use the values as: object1.rollno</p>
17.	Evaluate the syntax to create arrays in JSON.

JavaScript | JSON Arrays

The JSON Arrays is similar to JavaScript Arrays.

Syntax of Arrays in JSON Objects:

```
// JSON Arrays Syntax
```

```
{  
    "name": "Peter parker",  
    "heroName": "Spiderman",  
    "friends" : ["Deadpool", "Hulk", "Wolverine"]  
}
```

Accessing Array Values:

The Array values can be accessed using the index of each element in an Array.

EXAMPLE

```
<script>  
var myObj, i, x = "";  
myObj = {  
    "name": "John",  
    "age": 30,  
    "cars": [ "Ford", "BMW", "Fiat" ]  
};  
  
for (i in myObj.cars) {  
    x += myObj.cars[i] + "<br>";  
}  
document.getElementById("demo").innerHTML = x;  
</script>
```

OUTPUT

Looping through an array using a for in loop:

Ford
BMW
Fiat

18.	<p>How will you make a request with JSON?</p> <p>What is a JSON request?</p> <p>JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).</p> <p><u>jQuery getJSON() Method</u></p>
-----	---

The jQuery getJSON() method sends asynchronous http GET request to the server and retrieves the data in JSON format by setting accepts header to application/json, text/javascript. This is same as get() method, the only difference is that getJSON() method specifically retrieves JSON data whereas get() method retrieves any type of data. It is like shortcut method to retrieve JSON data.

```
$.getJSON(url,[data],[callback]);
```

Parameter Description:

- url: request url from which you want to retrieve the data
- data: JSON data to be sent to the server as a query string
- callback: function to be executed when request succeeds

The following example shows how to retrieve JSON data using getJSON() method.

Example: jQuery getJSON() Method

```
$.getJSON('/jquery/getjsondata', {name:'Steve'}, function (data, textStatus, jqXHR){  
    $('p').append(data.firstName);  
});  
  
<p></p>
```

OUTPUT

jQuery get() method demo

Steve

Define DDL and DML

DDL(Data Definition Language) : DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

Examples of DDL commands:

- **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- **DROP** – is used to delete objects from the database.
- **ALTER**-is used to alter the structure of the database.
- **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT** –is used to add comments to the data dictionary.
- **RENAME** –is used to rename an object existing in the database

19. Write SQL query to find minimum and maximum marks in a table.

DML(Data Manipulation Language) : The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

20.

	<p>Examples of DML:</p> <ul style="list-style-type: none"> • INSERT – is used to insert data into a table. • UPDATE – is used to update existing data within a table. • DELETE – is used to delete records from a database table. 		
	<p>PART - B</p>		
Q.No	<p>Questions</p> <p>i) Examine variables and data types in JavaScript.</p> <p style="text-align: center;">Variables in JavaScript:</p> <p>Variables in JavaScript are containers which hold reusable data. It is the basic unit of storage in a program.</p> <ul style="list-style-type: none"> • The value stored in a variable can be changed during program execution. • A variable is only a name given to a memory location, all the operations done on the variable effects that memory location. • In JavaScript, all the variables must be declared before they can be used. <p>JavaScript variables are containers for storing data values.</p> <p>In this example, <code>x</code>, <code>y</code>, and <code>z</code>, are variables:</p> <p>Examples</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 10px; vertical-align: top;"> <pre>var x = 5; var y = 6; var z = x + y;</pre> </td><td style="padding: 10px; vertical-align: top;"> <pre>var price1 = 5; var price2 = 6; var total = price1 + price2;</pre> </td></tr> </table> <p>1.</p> <h2>Declaring (Creating) JavaScript Variables</h2> <p>Creating a variable in JavaScript is called "declaring" a variable.</p> <p>You declare a JavaScript variable with the <code>var</code> keyword:</p> <pre>var carName;</pre> <p>After the declaration, the variable has no value (technically it has the value of <code>undefined</code>).</p> <p>To assign a value to the variable, use the equal sign:</p> <pre>carName = "Volvo";</pre> <p>You can also assign a value to the variable when you declare it:</p> <pre>var carName = "Volvo";</pre>	<pre>var x = 5; var y = 6; var z = x + y;</pre>	<pre>var price1 = 5; var price2 = 6; var total = price1 + price2;</pre>
<pre>var x = 5; var y = 6; var z = x + y;</pre>	<pre>var price1 = 5; var price2 = 6; var total = price1 + price2;</pre>		

EXAMPLE

```
<script>
var carName = "Volvo";
document.getElementById("demo").innerHTML
= carName;
</script>
```

OUTPUT

JavaScript Variables

Create a variable, assign a value to it, and display it:

Volvo

- ii) Give various Operators in JavaScript.

JavaScript Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description	Example
+	Addition	<code>var x = 5; var y = 2; var z = x + y;</code>
-	Subtraction	<code>var x = 5; var y = 2; var z = x - y;</code>
*	Multiplication	<code>var x = 5; var y = 2; var z = x * y;</code>
**	Exponentiation (ES2016)	
/	Division	
%	Modulus (Division Remainder)	
++	Increment	
--	Decrement	

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
----------	---------	---------

=	<code>x = y</code>	<code>x = y</code>
<code>+=</code>	<code>x += y</code>	<code>x = x + y</code>
<code>-=</code>	<code>x -= y</code>	<code>x = x - y</code>
<code>*=</code>	<code>x *= y</code>	<code>x = x * y</code>
<code>/=</code>	<code>x /= y</code>	<code>x = x / y</code>
<code>%=</code>	<code>x %= y</code>	<code>x = x % y</code>
<code>**=</code>	<code>x **= y</code>	<code>x = x ** y</code>

The **addition assignment** operator (`+=`) adds a value to a variable.

Assignment

```
var x = 10;
x += 5;
```

JavaScript String Operators

The `+` operator can also be used to add (concatenate) strings.

Example

```
var txt1 = "John";
var txt2 = "Doe";
var txt3 = txt1 + " " + txt2;
```

The result of `txt3` will be:

John Doe

The `+=` assignment operator can also be used to add (concatenate) strings:

Example

```
var txt1 = "What a very ";
txt1 += "nice day";
```

The result of `txt1` will be:

What a very nice day

When used on strings, the `+` operator is called the **concatenation operator**.

JavaScript Comparison Operators

Operator	Description
<code>==</code>	equal to
<code>====</code>	equal value and equal type
<code>!=</code>	not equal
<code>!==</code>	not equal value or not equal type

>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	Description
&&	logical and
	logical or
!	logical not

JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

Type operators are fully described in the [JS Type Conversion](#) chapter.

JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

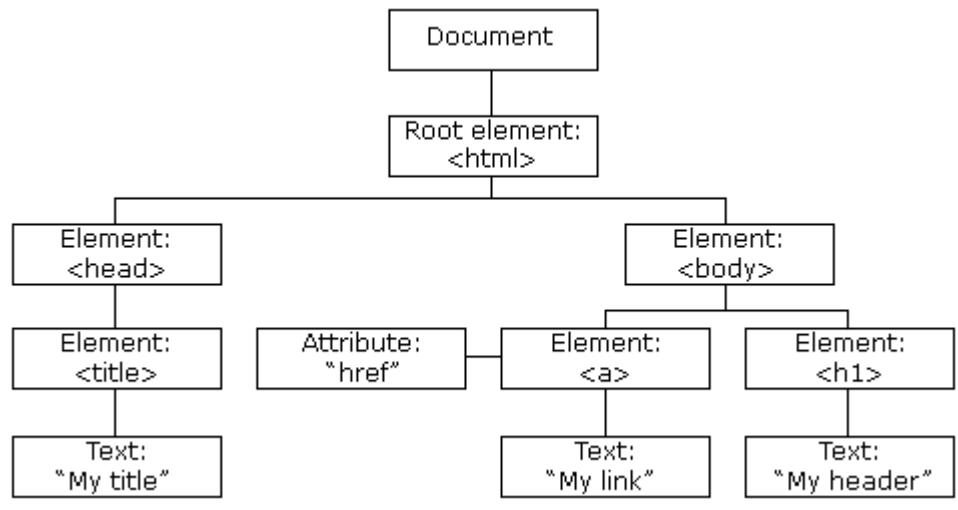
Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
	OR	5 1	0101 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2

	>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2
--	-----	-----------------------	---------	------------	------	---

The examples above uses 4 bits unsigned examples. But JavaScript uses 32-bit signed numbers. Because of this, in JavaScript, ~5 will not return 10. It will return -6.

~0000000000000000000000000000000101 will return 111111111111111111111111010

2.	<p>(i) Summarize about DOM Model.</p> <p>What is the DOM?</p> <p>The DOM is a W3C (World Wide Web Consortium) standard.</p> <p>The DOM defines a standard for accessing documents:</p> <p><i>"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."</i></p> <p>The W3C DOM standard is separated into 3 different parts:</p> <ul style="list-style-type: none"> • Core DOM - standard model for all document types • XML DOM - standard model for XML documents • HTML DOM - standard model for HTML documents <p><u>JavaScript HTML DOM</u></p> <p>With the HTML DOM, JavaScript can access and change all the elements of an HTML document.</p> <p>The HTML DOM (Document Object Model)</p> <p>When a web page is loaded, the browser creates a Document Object Model of the page.</p> <p>The HTML DOM model is constructed as a tree of Objects:</p> <p><u>The HTML DOM Tree of Objects</u></p>
----	--



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

(ii) **Describe** the concepts of Popup Boxes.

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box	Syntax	Example
<p>An alert box is often used if you want to make sure information comes through to the user.</p> <p>When an alert box pops up, the user will have to click "OK" to proceed</p>	<p>window.alert("sometext");</p> <p>The <code>window.alert()</code> method can be written without the window prefix</p>	<pre>alert("I am an alert box!");</pre>

	<p>Confirm Box</p> <p>When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.</p> <p>If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.</p>	<p>Syntax</p> <pre>window.confirm("sometext");</pre> <p>The window.confirm() method can be written without the window prefix.</p>	<p>Example</p> <pre>if(confirm("Press a button!")) { txt = "You pressed OK!"; } else { txt = "You pressed Cancel!"; }</pre>
	<p>Prompt Box</p> <p>A prompt box is often used if you want the user to input a value before entering a page.</p> <p>If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.</p>	<p>Syntax</p> <pre>window.prompt("sometext","defaultText");</pre> <p>The window.prompt() method can be written without the window prefix.</p>	<p>Example</p> <pre>var person = prompt("Please enter your name", "Harry Potter"); if(person == null person == "") { txt = "User cancelled the prompt."; } else { txt = "Hello " + person + "! How are you today?"; }</pre>
3.	<p>(i) Write short notes on Date and Objects.</p> <p>The Date Object. The Date object is a built-in object in JavaScript that stores the date and time. It provides a number of built-in methods for formatting and managing that data. By default, a new Date instance without arguments provided creates an object corresponding to the current date and time</p> <h3><u>The Date Object</u></h3> <p>The Date object is a built-in object in JavaScript that stores the date and time. It provides a number of built-in methods for formatting and managing that data.</p> <p>By default, a new Date instance without arguments provided creates an object corresponding to the current date and time. To demonstrate JavaScript's Date, let's create a variable and assign the current date to it.</p>		

EXAMPLE

now.js

```
// Set variable to current date and time
const now = new Date();

// View the output
now;
Output
Wed Oct 18 2017 12:41:34 GMT+0000 (UTC)
```

Date Creation	Output
<code>new Date()</code>	Current date and time
<code>new Date(timestamp)</code>	Creates date based on milliseconds since Epoch time
<code>new Date(date string)</code>	Creates date based on date string
<code>new Date(year, month, day, hours, minutes, seconds, milliseconds)</code>	Creates date based on specified date and time

(ii) **Explain** in detail about Regular expressions.

A JavaScript Regular Expression (or **Regex**) is a sequence of characters that we can utilize to work effectively with strings. Using this syntax, we can:

- **search** for text in a string
- **replace** substrings in a string
- **extract** information from a string

4.	<p>(i) Describe the control statements in Java.</p> <p>Conditional Statements</p> <p>Very often when you write code, you want to perform different actions for different decisions.</p>
----	---

You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

JavaScript Loops

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

```
<script>

var cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];

var text = "";

var i;
for (i = 0; i < cars.length; i++) {
    text += cars[i] + "<br>";

}

document.getElementById("demo").innerHTML = text;

</script>
```

OUTPUT

JavaScript For Loop

BMW
Volvo
Saab
Ford
Fiat
Audi

	(ii) Discuss any two validation functions in java script.
5.	<p>(i) Write a Java script to find the Prime number between 1 and 100.</p> <pre><html> <head> <title>JavaScript Prime</title> </head> <body> <script> for (var limit = 1; limit <= 20; limit++) { var a = false; for (var i = 2; i <= limit; i++) { if (limit%i==0 && i!=limit) { a = true; } } if (a === false) { document.write("
" + limit); } } </script> </body> </html></pre> <p>(ii) Write a Java Script to find factorial of a given number.</p> <pre>function factorial(n) { return (n != 1) ? n * factorial(n - 1) : 1; } alert(factorial(5)); // 120</pre>
6.	<p>(i) Demonstrate a java script for displaying the context menu.</p> <p>(ii) Demonstrate a java script to display the welcome message using the alert whenever button of a html form is pressed.</p> <pre><html> <head> <title>Display Alert Message on Button Click Event.</title> <script type="text/javascript"> function showMessage(){ alert("Welcome friends, You pressed the Button."); } </script> </head> <body></pre>

```

<center>
    <h1>Display Alert Message on Button Click Event.</h1>
    <b>Click on button to display message:</b><br><br>
    <input type="button" id="btnShowMsg" value="Click Me!">
    onClick='showMessage()'
</center>
</body>
</html>

```

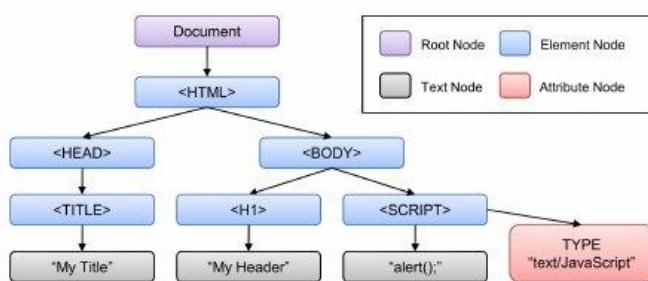
Result

Welcome friends, You pressed the Button.

- (i) Evaluate how DOM Nodes and Trees can be used.

What is the purpose of HTML DOM Node Tree?

- HTML DOM view the HTML document with a tree structure format and it consists of root node and child nodes.
- The node-tree is being accessed using the tree formation and the structure in which the elements get created.
- The contents that are used being modified or removed using the new elements and it can be created within the limitations.
- The structure consists of a document that is the root and within it Root element <html> from where the tree starts.
- It consists of sub-elements like <head> and <body> and other text and attributes written in the HTML format.



- (ii) Evaluate the way of Traversing and modifying a DOM Tree.

8. (i) **Discuss the concepts of Registering Event handlers.**
As mentioned above, **events** are actions or occurrences that happen in the system you are programming — the system produces (or "fires") a signal of some kind when an event occurs,

and also provides a mechanism by which some kind of action can be automatically taken (that is, some code running) when the event occurs. For example in an airport when the runway is clear for a plane to take off, a signal is communicated to the pilot, and as a result, they commence piloting the plane.

In the case of the Web, events are fired inside the browser window, and tend to be attached to a specific item that resides in it — this might be a single element, set of elements, the HTML document loaded in the current tab, or the entire browser window. There are a lot of different types of events that can occur, for example:

The user clicking the mouse over a certain element or hovering the cursor over a certain element.

The user pressing a key on the keyboard.

The user resizing or closing the browser window.

A web page finishing loading.

A form being submitted.

A video being played, or paused, or finishing play.

An error occurring.

Each available event has an **event handler**, which is a block of code (usually a JavaScript function that you as a programmer create) that will be run when the event fires. When such a block of code is defined to be run in response to an event firing, we say we are **registering an event handler**. Note that event handlers are sometimes called **event listeners** — they are pretty much interchangeable for our purposes, although strictly speaking, they work together. The listener listens out for the event happening, and the handler is the code that is run in response to it happening.

In the following example, we have a single `<button>`, which when pressed, makes the background change to a random color:

```
<button>Change color</button>
```

The JavaScript looks like so:

```
const btn = document.querySelector('button');

function random(number) {
    return Math.floor(Math.random() * (number+1));
}

btn.onclick = function() {
    const rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' +
random(255) + ')';
    document.body.style.backgroundColor = rndCol;
}
```

(ii) **Discuss** the concepts of Event Handling.

JavaScript Events

HTML events are "**things**" that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can "**react**" on these events.

HTML Events

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

With single quotes:

`<element event='some JavaScript'>`

With double quotes:

`<element event="some JavaScript">`

In the following example, an `onclick` attribute (with code), is added to a `<button>` element:

Example

```
<button onclick="document.getElementById('demo').innerHTML = Date()">The time  
is?</button>
```

OUTPUT

Sun Jan 12 2020 06:00:41 GMT+0530 (India Standard Time)

In the example above, the JavaScript code changes the content of the element with id="demo".

9. **Analyze** a web page to create a clock with timing event.
- ```
function showTime(){
 var date = new Date();
 var h = date.getHours(); // 0 - 23
 var m = date.getMinutes(); // 0 - 59
```

```
var s = date.getSeconds(); // 0 - 59
var session = "AM";

if(h == 0){
 h = 12;
}

if(h > 12){
 h = h - 12;
 session = "PM";
}

h = (h < 10) ? "0" + h : h;
m = (m < 10) ? "0" + m : m;
s = (s < 10) ? "0" + s : s;

var time = h + ":" + m + ":" + s + " " + session;
document.getElementById("MyClockDisplay").innerText = time;
document.getElementById("MyClockDisplay").textContent = time;

setTimeout(showTime, 1000);

}

showTime();
```

---

```
<html>
<head>
<title>Digital Clock</title>
<style>
#clock{
 color:#F0F;
}
</style>
</head>
<body>
<script>
function digclock()
{
 var d = new Date()
 var t = d.toLocaleTimeString()
```

```

 document.getElementById("clock").innerHTML = t
 }

 setInterval(function(){digclock()},1000)

</script>
Digital Clock
<div id="clock">

</div>
</body>
</html>

```

**OUTPUT**

**Output :**

Digital Clock  
**5:54:26 PM**

10. (i) **Write short notes on DHTML with JavaScript.**
- Dynamic HyperText Markup Language (DHTML) is a combination of Web development technologies used to create dynamically changing websites. Web pages may include animation, dynamic menus and text effects. The technologies used include a combination of HTML, JavaScript or VB Script, CSS and the document object model (DOM).
- Designed to enhance a Web user's experience, DHTML includes the following features:
- Dynamic content, which allows the user to dynamically change Web page content
  - Dynamic positioning of Web page elements
  - Dynamic style, which allows the user to change the Web page's color, font, size or content
- EXAMPLE**
- ```

<!DOCTYPE html PUBLIC "-//abc//DTD XHTML 1.1//EN"
"http://www.abc.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.abc.org/1999/xhtml">
<head>
<title>DHTML example</title>
<script type="text/JavaScript">
    function greet_user()
    {
        var name=document.getElementById("userName").value;

```

```

if(name=="")
{
    alert("Welcome"+name);
}
else
{
    alert("Please provide User Name")
}

```

</script>

</head>

<body>

<table border="1" cellspacing="3">

<tr>

<td colspan="2"><h6> Please Enter Your Name </h6></td>

</tr>

<tr>

<td><h4>User Name </h4></td>

<td><input type="text" id="userName" ></td>

</tr>

<tr>

<td colspan="2"><input type="button" value="Submit"

onclick="greet_user()"/>

</td>

</tr>

</table>

</body>

</html>

(ii) **Classify** moving elements.

Explain the concept of JSON with example.

JSON is a format for storing and transporting data.

JSON is often used when data is sent from a server to a web page.

What is JSON?

11.

- JSON stands for **JavaScript Object Notation**
- JSON is a lightweight data interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand

* The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language.

JSON Example

This JSON syntax defines an employees object: an array of 3 employee records (objects):

JSON Example

```
{  
  "employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
  ]  
}
```

Describe in detail about JSON Objects and Arrays.

JSON Arrays

Arrays as JSON Objects

Example

```
[ "Ford", "BMW", "Fiat" ]
```

Arrays in JSON are almost the same as arrays in JavaScript.

In JSON, array values must be of type string, number, object, array, boolean or *null*.

In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and *undefined*.

12.

Arrays in JSON Objects

Arrays can be values of an object property:

Example

```
{  
  "name": "John",  
  "age": 30,  
  "cars": [ "Ford", "BMW", "Fiat" ]  
}
```

Accessing Array Values

You access the array values by using the index number:

Example

```
x = myObj.cars[0];
```

Looping Through an Array

You can access array values by using a **for-in** loop:

Example

```
for (i in myObj.cars) {  
    x += myObj.cars[i];  
}
```

OUTPUT

Looping through an array using a for in loop:

```
Ford  
BMW  
Fiat
```

Analyze about Function files and Http Request with sample program.

```
<!doctype html>  
<title>Example</title>  
  
<script>  
// Store XMLHttpRequest and the JSON file location in variables  
var xhr = new XMLHttpRequest();  
var url = "https://www.quackit.com/json/tutorial/artists.txt";  
  
// Called whenever the readyState attribute changes  
xhr.onreadystatechange = function() {  
  
    // Check if fetch request is done  
    if (xhr.readyState == 4 && xhr.status == 200) {  
  
        // Parse the JSON string  
        var jsonData = JSON.parse(xhr.responseText);  
  
        // Call the showArtists(), passing in the parsed JSON string  
        showArtists(jsonData);  
    }  
};
```

```

// Do the HTTP call using the url variable we specified above
xhr.open("GET", url, true);
xhr.send();

// Function that formats the string with HTML tags, then outputs the result
function showArtists(data) {
    var output = "<ul>"; // Open list
    var i;

    // Loop through the artists, and add them as list items
    for (var i in data.artists) {
        output += "<li>" + data.artists[i].artistname + " (Born: " + data.artists[i].born + ")</li>";
    }

    output += "</ul>"; // Close list

    // Output the data to the "artistlist" element
    document.getElementById("artistList").innerHTML = output;
}

</script>

<!-- The output appears here -->
<div id="artistList"></div>

```

OUTPUT

- Leonard Cohen (Born: 1934)
- Joe Satriani (Born: 1956)
- Snoop Dogg (Born: 1971)

| | |
|-----|---|
| 14. | <p>Summarize about</p> <p>(i) SQL Data Definition Commands</p> <p>Examples of Sql DDL commands are</p> <ul style="list-style-type: none"> • CREATE – Create an object. I mean, <u>create a database</u>, <u>table</u>, <u>triggers</u>, index, <u>functions</u>, <u>stored procedures</u>, etc. • DROP – This SQL DDL command helps to delete objects. For example, delete tables, delete a database, etc. • <u>ALTER</u> – Used to alter the existing database or its object structures. • <u>TRUNCATE</u> – This SQL DDL command removes records from tables • <u>RENAME</u> – Renaming the database objects <p>(ii) Data Manipulation Commands.</p> <p>Examples of DML commands are</p> <ul style="list-style-type: none"> • <u>SELECT</u> – This SQL DML command select records or data from a table • <u>INSERT</u> – Insert data into a database table. • <u>UPDATE</u> – This SQL DML command will update existing records within a table |
|-----|---|

| | |
|-----------------|--|
| | <ul style="list-style-type: none"> • <u>DELETE</u> – Delete unwanted records from a table |
| PART – C | |
| Q.No | Questions |
| 1. | <p>Analyze the merits and demerits of DOM parser with neat example.</p> <pre> <!DOCTYPE html> <html> <head> <script type="text/javascript"> var str = "<root><customer name='John' address='Chicago'></customer></root>"; function CreateXMLDocument () { var xmlDoc = null; if (window.DOMParser) { var parser = new DOMParser (); xmlDoc = parser.parseFromString (str, "text/xml"); } else if (window.ActiveXObject) { xmlDoc = new ActiveXObject ("Microsoft.XMLDOM"); xmlDoc.async = false; xmlDoc.loadXML (str); } var customerNode = xmlDoc.getElementsByTagName ("customer")[0]; var customerName = customerNode.getAttribute ("name"); alert ("The name of the first customer is " + customerName); } </script> </head> <body> <button onclick="CreateXMLDocument ();">Create an XML document from a string</button> </body> </html></pre> |
| 2. | <p>Consider a java script and HTML to create a page with two panes. The first pane (on left) should have a text area where HTML code can be typed by the user. The pane on the left should have a text area where HTML code can be typed by the user. The pane on the right side should display the preview of the HTML code typed by the user, as it would be seen on the browser.</p> |
| 3. | <p>Develop a DHTML program to handle the user click event.</p> <pre> <!DOCTYPE html> <html></pre> |

| | |
|----|---|
| | <pre><body> <p>Click the button to display the time.</p> <button onclick="getElementById('demo').innerHTML=Date()">What is the time?</button> <p id="demo"></p> </body> </html></pre> <p>OUTPUT</p> <p>Click the button to display the time.</p> <p>What is the time?</p> <p>Sat Jan 11 2020 22:36:24 GMT+0530 (India Standard Time)</p> |
| 4. | <p>Formulate a JavaScript program that work with JSON.</p> <p>JSON Object Syntax</p> <p>Example</p> <pre>{ "name":"John", "age":30, "car":null }</pre> <ul style="list-style-type: none"> ➤ JSON objects are surrounded by curly braces {}. ➤ JSON objects are written in key/value pairs. ➤ Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null). ➤ Keys and values are separated by a colon. ➤ Each key/value pair is separated by a comma. <p>Accessing Object Values</p> <p>You can access the object values by using dot (.) notation:</p> <p>Example</p> <pre>myObj = { "name":"John", "age":30, "car":null }; x = myObj.name;</pre> <p>You can also access the object values by using bracket ([]) notation:</p> <p>Example</p> |

```
myObj = { "name":"John", "age":30, "car":null };
x = myObj["name"];
```

Looping an Object

You can loop through object properties by using the for-in loop:

Example

In a for-in loop, use the bracket notation to access the property *values*:

Example

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

<p>How to loop through all properties in a JSON object.</p>

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

<script>
var myObj, x;
myObj = {"name":"John", "age":30, "car":null};
for (x in myObj) {
  document.getElementById("demo").innerHTML += x + " " + myObj[x] + "<br>";
}
</script>

</body>
</html>
```

OUTPUT

How to loop through all properties in a JSON object.

```
name John
age 30
car null
```

Nested JSON Objects

Values in a JSON object can be another JSON object.

Example

```
myObj = {
  "name":"John",
```

```
"age":30,  
"cars": {  
    "car1":"Ford",  
    "car2":"BMW",  
    "car3":"Fiat"  
}  
}
```

You can access nested JSON objects by using the dot notation or bracket notation:

Example

```
x = myObj.cars.car2;  
// or:  
x = myObj.cars["car2"];
```

OUTPUT

How to access nested JSON objects.

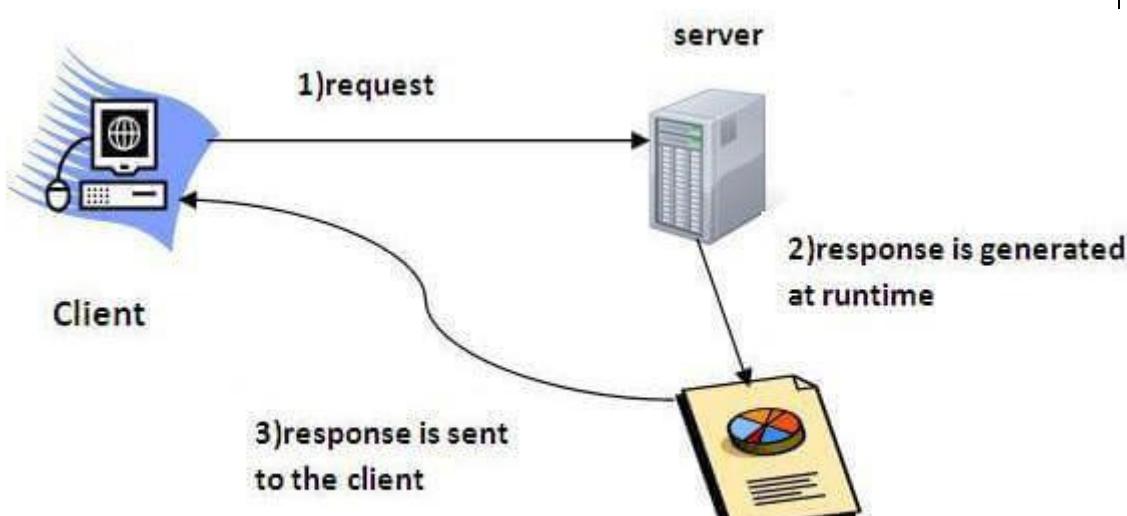
BMW
BMW

CS8651 Internet Programming – 2017Reg

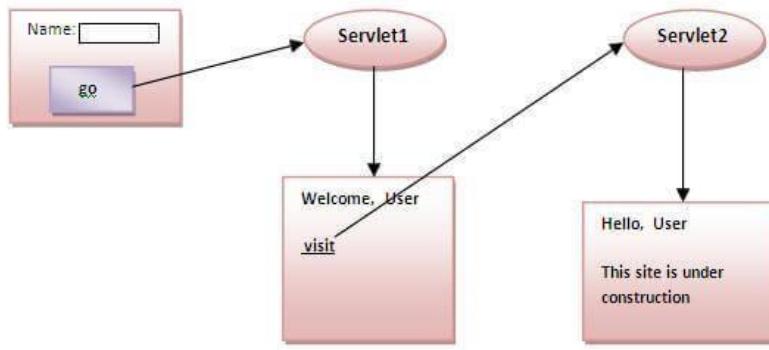
**UNIT III
- SERVER SIDE PROGRAMMING**

Servlets: Java Servlet Architecture – Servlet Life Cycle – Form GET and POST actions – Session Handling – Understanding Cookies – Installing and Configuring Apache Tomcat Web Server; **DATABASE CONNECTIVITY:** JDBC perspectives, JDBC program example; **JSP:** Understanding Java Server Pages – JSP Standard Tag Library (JSTL) – Creating HTML forms by embedding JSP code.

PART-A

Q.No	Questions
1.	<p>What are servlets?</p> <p>A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.</p> <ul style="list-style-type: none"> ➤ Servlet is a technology which is used to create a web application. ➤ Servlet is an API that provides many interfaces and classes including documentation. ➤ Servlet is an interface that must be implemented for creating any Servlet. ➤ Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests. ➤ Servlet is a web component that is deployed on the server to create a dynamic web page. 
2.	<p>List the application of servlets.</p> <p>Servlets may be used at different levels on a distributed framework. The following are some examples of servlet usage:</p> <ul style="list-style-type: none"> ➤ To accept form input and generate HTML Web pages dynamically.

	<ul style="list-style-type: none"> ➤ As part of middle tiers in enterprise networks by connecting to SQL databases via JDBC. ➤ In conjunction with applets to provide a high degree of interactivity and dynamic Web content generation. ➤ For collaborative applications such as online conferencing. ➤ A community of servlets could act as active agents which share data with each other. ➤ Servlets could be used for balancing load among servers which mirror the same content. ➤ Protocol support is one of the most viable uses for servlets. For example, a file service can start with NFS and move on to as many protocols as desired; the transfer between the protocols would be made transparent by servlets. Servlets could be used for tunneling over HTTP to provide chat, newsgroup or other file server functions.
3.	<p>Summarize the advantages and disadvantages of servlets.</p> <p>Servlet Advantage</p> <ul style="list-style-type: none"> ➤ Servlets provide a way to generate dynamic documents that is both easier to write and faster to run. ➤ Provide all the powerfull features of JAVA, such as Exception handling and garbage collection. ➤ Servlet enables easy portability across Web Servers. ➤ Servlet can communicate with different servlet and servers. ➤ Since all web applications are stateless protocol, servlet uses its own API to maintain session <p>Servlet Disadvantage</p> <ul style="list-style-type: none"> ➤ Designing in servlet is difficult and slows down the application. ➤ Writing complex business logic makes the application difficult to understand. ➤ You need a Java Runtime Environment on the server to run servlets. ➤ CGI is a completely language independent protocol, so you can write CGIs in whatever languages you have available (including Java if you want to).
4.	<p>Show how is session tracking is achieved by the URL rewriting?</p> <p>URL Rewriting</p> <p>In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format:</p> <p>url?name1=value1&name2=value2&??</p> <p>A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.</p>



Advantage of URL Rewriting

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

1. It will work only with links.
2. It can send Only textual information.

URL Rewriting

URL rewriting is another way to support anonymous session tracking. With URL rewriting, every local URL the user might click on is dynamically modified, or rewritten, to include extra information. The extra information can be in the form of extra path information, added parameters, or some custom, server-specific URL change.

Example 7-2 shows a revised version of our shopping cart viewer that uses URL rewriting in the form of extra path information to anonymously track a shopping cart.

```

Example 7-2. Session tracking using URL rewriting
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ShoppingCartViewerRewrite extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
        ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("");
        out.println("");
        // Get the current session ID, or generate one if necessary
        String sessionid = req.getPathInfo();
        if (sessionid == null)
            sessionid = generateSessionId();
    }
}

```

```

}

// Cart items are associated with the session ID
String[] items = getItemsFromCart(sessionid);
// Print the current cart items.
out.println("You currently have the following items in your cart:
");
if (items == null) { out.println("None"); } else { out.println(
    );
    for (int i = 0; i < items.length; i++)
    { out.println(
        • " + items[i]); } out.println(
    );
}

// Ask if the user wants to add more items or check out. // Include the session
ID in the action URL.
out.println(
");
out.println("Would you like to
");
out.println(" \\""); out.println(" \\""); out.println(
");

// Offer a help page. Include the session ID in the URL. out.println("For
help, click here");
out.println(""); }

private static String generateSessionId() { String uid = new
java.rmi.server.UID().toString();

// guaranteed unique return java.net.URLEncoder.encode(uid);

// encode any special chars }

private static String[] getItemsFromCart(String sessionid) {

// Not implemented }

}

This servlet first tries to retrieve the current session ID using
getPathInfo() . If a session ID is not specified, it calls
generateSessionId() to generate a new unique session ID using an RMI

```

	<p>class designed specifically for this. The session ID is used to fetch and display the current items in the cart. The ID is then added to the form's ACTION attribute, so it can be retrieved by the ShoppingCart servlet. The session ID is also added to a new help URL that invokes the Help servlet. This wasn't possible with hidden form fields because the Help servlet isn't the target of a form submission. docstore.mik.ua/oreilly/javaent/servlet/ch07_03.htm</p>												
5.	<p>Compare GET and POST request type.</p> <table border="1"> <thead> <tr> <th>GET</th><th>POST</th></tr> </thead> <tbody> <tr> <td>1) In case of Get request, only limited amount of data can be sent because data is sent in header.</td><td>In case of post request, large amount of data can be sent because data is sent in body.</td></tr> <tr> <td>2) Get request is not secured because data is exposed in URL bar.</td><td>Post request is secured because data is not exposed in URL bar.</td></tr> <tr> <td>3) Get request can be bookmarked.</td><td>Post request cannot be bookmarked.</td></tr> <tr> <td>4) Get request is idempotent. It means second request will be ignored until response of first request is delivered</td><td>Post request is non-idempotent.</td></tr> <tr> <td>5) Get request is more efficient and used more than Post.</td><td>Post request is less efficient and used less than get.</td></tr> </tbody> </table>	GET	POST	1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.	2) Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.	3) Get request can be bookmarked.	Post request cannot be bookmarked.	4) Get request is idempotent . It means second request will be ignored until response of first request is delivered	Post request is non-idempotent .	5) Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.
GET	POST												
1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.												
2) Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.												
3) Get request can be bookmarked.	Post request cannot be bookmarked.												
4) Get request is idempotent . It means second request will be ignored until response of first request is delivered	Post request is non-idempotent .												
5) Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.												
6.	<p>Summarize the servlet interface and its methods.</p> <p>Servlet Interface</p> <ol style="list-style-type: none"> 1. Servlet Interface 2. Methods of Servlet interface <ul style="list-style-type: none"> ➤ Servlet interface provides common behavior to all the servlets. ➤ Servlet interface defines methods that all servlets must implement. 												

- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).
- It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

Servlet Example by implementing Servlet interface

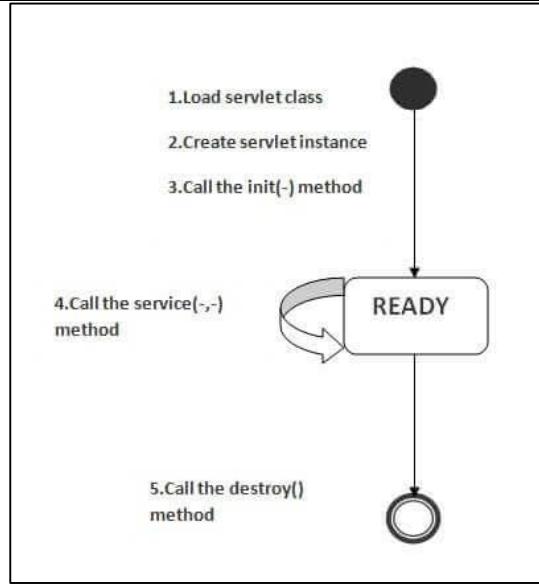
File: First.java

```

1. import java.io.*;
2. import javax.servlet.*;
3.
4. public class First implements Servlet{
5.     ServletConfig config=null;
6.
7.     public void init(ServletConfig config){
8.         this.config=config;
9.         System.out.println("servlet is initialized");
10.    }

```

	<pre> 11. 12. public void service(ServletRequest req,ServletResponse res) 13. throws IOException,ServletException{ 14. 15. res.setContentType("text/html"); 16. 17. PrintWriter out=res.getWriter(); 18. out.print("<html><body>"); 19. out.print("hello simple servlet"); 20. out.print("</body></html>"); 21. 22. } 23. public void destroy(){System.out.println("servlet is destroyed");} 24. public ServletConfig getServletConfig(){return config;} 25. public String getServletInfo(){return "copyright 2007-1010";} 26. 27. }</pre>
7.	<p>Sketch the Servlet life cycle.</p> <h3>Life Cycle of a Servlet (Servlet Life Cycle)</h3> <p>The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:</p> <ol style="list-style-type: none"> A. <u>Life Cycle of a Servlet</u> <ol style="list-style-type: none"> 1. <u>Servlet class is loaded</u> 2. <u>Servlet instance is created</u> 3. <u>init method is invoked</u> 4. <u>service method is invoked</u> 5. <u>destroy method is invoked</u>



As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

Show the use of 'param' variable in JSP.

Jsp param

<jsp:param> tag is used to represent parameter value during jsp forward or include action this should be the sub tag of **<jsp:forward>** or **<jsp:include>**.

When an include or forward element is invoked, the original request object is provided to the target page. If you wish to provide additional data to that page, you can append parameters to the request object by using the **jsp:param** element.

8.

Syntax

```
<jsp:param name=" " value=" "/>
```

Example

```
<jsp:include page="contact.jsp"/>
<jsp:param name="param1" value="value1"/>
</jsp:include>
```

Example

```
<jsp:forward page="home.jsp"/>
```

	<pre><jsp:param name="param1" value="value1"/> </jsp:forward></pre>
	<p>Quote the uses of cookies.</p> <p><u>Cookies in Servlet</u></p> <p>A cookie is a small piece of information that is persisted between the multiple client requests.</p> <p>A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.</p> <p>How Cookie works</p> <p>By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.</p> <p>Types of Cookie</p> <p>There are 2 types of cookies in servlets.</p> <p>9.</p> <ol style="list-style-type: none"> 1. Non-persistent cookie 2. Persistent cookie <p>Non-persistent cookie</p> <p>It is valid for single session only. It is removed each time when user closes the browser.</p> <p>Persistent cookie</p> <p>It is valid for multiple session. It is not removed each time when user closes the browser. It is removed only if user logout or signout.</p> <p>Advantage of Cookies</p> <ol style="list-style-type: none"> 1. Simplest technique of maintaining the state. 2. Cookies are maintained at client side. <p>Disadvantage of Cookies</p> <ol style="list-style-type: none"> 1. It will not work if cookie is disabled from the browser.

	2. Only textual information can be set in Cookie object.
10.	<p>Analyze about java scriptlet.</p> <p>In JavaServer Pages (JSP) technology, a scriptlet is a piece of Java-code embedded in the HTML-like JSP code. The scriptlet is everything inside the <code><% %></code> tags. Between these the user can add any valid Scriptlet i.e. any valid Java Code. In AppleScript, a scriptlet is a small script.</p> <h3>JSP scriptlet tag</h3> <p>A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:</p> <pre><% java source code %></pre> <h3>Example of JSP scriptlet tag</h3> <p>In this example, we are displaying a welcome message.</p> <pre><html> <body> <% out.print("welcome to jsp"); %> </body> </html></pre>
11.	<p>Express appropriate java script code to remove an element (current element) from a DOM.</p> <p>A. Removing Existing HTML Elements</p> <p>To remove an HTML element, use the <code>remove()</code> method:</p> <p style="text-align: center;">1. Example</p> <pre><div> <p id="p1">This is a paragraph.</p> <p id="p2">This is another paragraph.</p> </div> <script> var elmnt = document.getElementById("p1"); elmnt.remove(); </script></pre>
12.	Define JSP.

	<p>JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.</p> <p>A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.</p> <p>Using JSP, you can collect input from users through Webpage forms, present records from a database or another source, and create Webpages dynamically.</p> <p>JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages, and sharing information between requests, pages etc.</p>
13.	<p>List any three advantages of java servlet over JSP.</p> <ol style="list-style-type: none"> 1. Being an extension to <u>Java servlet</u>, it can use every feature of Java Servlet. Also, custom tags can be used along with it. 2. There is no need to recompile JSP when changed. The changes automatically appear when run. 3. The tags which are used are easy to understand and write. 4. Supports Java API's which can now be easily used and integrated with the HTML code. 5. The results which are obtained are in HTML format, so can be opened on any browsers. 6. Customized JSP tags can be used. Ex: Tags with XML. 7. Changes can be added into the business logic page rather than changing in each and every page.
14.	<p>Rewrite the code segment to store current server time in session using Java Servlet API.</p> <p>Write a servlet application to print the current date and time.</p> <p>Answer:</p> <p>The most important advantage of using Servlet is that we can use all the</p>

methods available in core java. The Date class is available in java.util package.

Below program shows how to print the current date and time. We can use simple Date object with `toString()` to print current date and time.

DateSrv.java

```
import java.io.*;
import javax.servlet.*;

public class DateSrv extends GenericServlet
{
    //implement service()
    public void service(ServletRequest req, ServletResponse res) throws
IOException, ServletException
    {
        //set response content type
        res.setContentType("text/html");
        //get stream obj
        PrintWriter pw = res.getWriter();
        //write req processing logic
        java.util.Date date = new java.util.Date();
        pw.println("<h2>"+"Current Date & Time: "
+date.toString()+"</h2>");
        //close stream object
        pw.close();
    }
}
```

Output:

Current Date & Time: Mon Dec 12 18:01:39 IST 2016

15.

Compare the difference between JSP and servlet.

Difference Between Servlet and JSP

In this article we will list some of the differences between Servlets and JSP.

SERVLET	JSP
A servlet is a server-side program and written purely on Java.	JSP is an interface on top of Servlets. In another way, we can say that JSPs are extension of servlets to minimize the effort of developers to write User Interfaces using Java programming.
Servlets run faster than JSP	JSP runs slower because it has the transition phase for converting from JSP page to a Servlet file. Once it is converted to a Servlet then it will start the compilation
Executes inside a Web server, such as Tomcat	A JSP program is compiled into a Java servlet before execution. Once it is compiled into a servlet, its life cycle will be same as of servlet. But, JSP has its own API for the lifecycle.
Receives HTTP requests from users and provides HTTP responses	Easier to write than servlets as it is similar to HTML.
We can not build any custom tags	One of the key advantages is we can build custom tags using JSP API (there is a separate

		package available for writing the custom tags which can be available as the re-usable components with lot of flexibility
	<p>Servlet advantages include:</p> <p>1. Performance : get loaded upon first request and remains in memory idenfinitely.</p> <p>2. Simplicity : Run inside controlled server environment. No specific client software is needed:web broser is enough</p> <p>3. Session Management : overcomes HTTP's stateless nature</p> <p>4. Java Technology : network access,Database connectivity, j2ee integration</p>	<p>JSP Provides an extensive infrastructure for:</p> <ul style="list-style-type: none"> 1. Tracking sessions. 2. Managing cookies. 3. Reading and sending HTML headers. 4. Parsing and decoding HTML form data. <p>5. JSP is Efficient: Every request for a JSP is handled by a simple Java thread</p> <p>6. JSP is Scalable: Easy integration with other backend services</p> <p>7. Separation of roles: Developers, Content Authors/Graphic Designers/Web Masters</p>
<h2>II. Difference between Servlet and JSP</h2> <p>In brief, it can be defined as Servlet are the java programs that run on a Web server and act as a middle layer between a request coming from HTTP client and databases or applications on the HTTP server. While JSP is simply a text document that contains two types of text: static text which is predefined and dynamic text which is rendered after server response is received.</p>		

sr. No.	Key	Servlet	JSP
1	Implementation	Servlet is developed on Java language.	JSP is primarily written in HTML language although Java code could also be written on it but for it, JSTL or other language is required.
2	MVC	In contrast to MVC we can state servlet as a controller which receives the request process and send back the response.	On the other hand, JSP plays the role of view to render the response returned by the servlet.
3	Request type	Servlets can accept and process all type of protocol requests.	JSP on the other hand is compatible with HTTP request only.
4	Session Management	In Servlet by default session management is not enabled, the user has to enable it explicitly.	On the other hand in JSP session management is automatically enabled.
5	Performance	Servlet is faster than JSP.	JSP is slower than Servlet because first the translation of JSP to java code is taking place and then compiles.
6	Modification reflected	Modification in Servlet is a time-consuming task because it includes reloading, recompiling and restarting the server as we made any change in our code to get reflected.	On the other hands JSP modification is fast as just need to click the refresh button and code change would get reflected.

Summarize briefly about the interaction between a webserver and a servlet.

How does a web server interact with a servlet?

A servlet is a [Java Programming](#) language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. It is also a web component that is deployed on the server to create a dynamic web page.

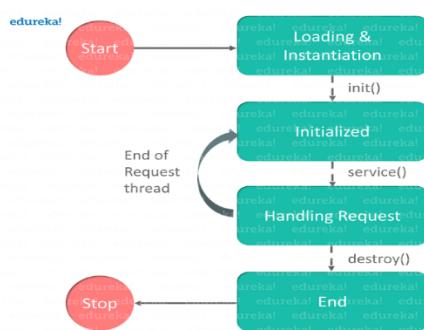
16.



In this figure you can see, a client sends a request to the server and the server generates the response, analyses it and sends the response to the client.

Stages of the Servlet Life Cycle: The Servlet life cycle mainly goes through four stages,

- Loading a Servlet.
- Initializing the Servlet.
- Request handling
- Destroying the Servlet.



	<p>Let's look at each of these stages in details:</p> <ol style="list-style-type: none"> 1. Loading a Servlet: The first stage of the Servlet life cycle involves loading and initializing the Servlet by the Servlet container. The Web container or Servlet Container can load the Servlet at either of the following two stages :Initializing the context, on configuring the Servlet with a zero or positive integer value.If the Servlet is not preceding the stage, it may delay the loading process until the Web container determines that this Servlet is needed to service a request. 2. Initializing a Servlet: After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object. The container initializes the Servlet object by invoking the <i>init(ServletConfig)</i> method which accepts ServletConfig object reference as a parameter. 3. Handling request: After initialization, the Servlet instance is ready to serve the client requests. The Servlet container performs the following operations when the Servlet instance is located to service a request :It creates the ServletRequest and ServletResponse. In this case, if this is an HTTP request then the Web container creates HttpServletRequest and HttpServletResponse objects which are subtypes of the ServletRequest and ServletResponse objects respectively. 4. Destroying a Servlet: When a Servlet container decides to destroy the Servlet, it performs the following operations,It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.After currently running threads have completed their jobs, the Servlet container calls the destroy() method on the Servlet instance. <p>After the destroy() method is executed, the Servlet container releases all the references of this Servlet instance so that it becomes eligible for garbage collection.</p>
17.	<p>Define JDBC.</p> <h2>What is JDBC?</h2> <p>JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.</p> <p>The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.</p> <ul style="list-style-type: none"> • Making a connection to a database. • Creating SQL or MySQL statements. • Executing SQL or MySQL queries in the database. • Viewing & Modifying the resulting records.

	<p>Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as –</p> <ul style="list-style-type: none"> • Java Applications • Java Applets • Java Servlets • Java ServerPages (JSPs) • Enterprise JavaBeans (EJBs). <p>All of these different executables are able to use a JDBC driver to access a database, and take advantage of the stored data.</p> <p>JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.</p>
	<p>Formulate the three methods that are central to the life cycle of the servlet.</p> <h2 style="text-align: center;">Servlets - Life Cycle</h2> <p>A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.</p> <ul style="list-style-type: none"> • The servlet is initialized by calling the init() method. • The servlet calls service() method to process a client's request. • The servlet is terminated by calling the destroy() method. • Finally, servlet is garbage collected by the garbage collector of the JVM. <p>Now let us discuss the life cycle methods in detail.</p> <h3>18. The init() Method</h3> <p>The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.</p> <p>The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.</p> <p>When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.</p> <p>The init method definition looks like this –</p> <pre style="background-color: #f0f0f0; padding: 10px;"><code>public void init() throws ServletException { // Initialization code... }</code></pre>

The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

A. The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this –

```
public void destroy() {  
    // Finalization code...  
}
```

19.	<p>Distinguish between servlets and JSP.</p> <p>Difference between Servlet and JSP</p> <table border="1" data-bbox="354 1298 1576 1983"><thead><tr><th data-bbox="354 1298 849 1365">SERVLET</th><th data-bbox="849 1298 1576 1365">JSP</th></tr></thead><tbody><tr><td data-bbox="354 1365 849 1432">Servlet is a java code.</td><td data-bbox="849 1365 1576 1432">JSP is a html based code.</td></tr><tr><td data-bbox="354 1432 849 1545">Writing code for servlet is harder than JSP as it is html in java.</td><td data-bbox="849 1432 1576 1545">JSP is easy to code as it is java in html.</td></tr><tr><td data-bbox="354 1545 849 1657">Servlet plays a controller role in MVC approach.</td><td data-bbox="849 1545 1576 1657">JSP is the view in MVC approach for showing output.</td></tr><tr><td data-bbox="354 1657 849 1792">Servlet is faster than JSP.</td><td data-bbox="849 1657 1576 1792">JSP is slower than Servlet because the first step in JSP lifecycle is the translation of JSP to java code and then compile.</td></tr><tr><td data-bbox="354 1792 849 1904">Servlet can accept all protocol requests.</td><td data-bbox="849 1792 1576 1904">JSP only accept http requests.</td></tr><tr><td data-bbox="354 1904 849 1983">In Servlet, we can override the service() method.</td><td data-bbox="849 1904 1576 1983">In JSP, we cannot override its service() method.</td></tr></tbody></table>	SERVLET	JSP	Servlet is a java code.	JSP is a html based code.	Writing code for servlet is harder than JSP as it is html in java.	JSP is easy to code as it is java in html.	Servlet plays a controller role in MVC approach.	JSP is the view in MVC approach for showing output.	Servlet is faster than JSP.	JSP is slower than Servlet because the first step in JSP lifecycle is the translation of JSP to java code and then compile.	Servlet can accept all protocol requests.	JSP only accept http requests.	In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.
SERVLET	JSP														
Servlet is a java code.	JSP is a html based code.														
Writing code for servlet is harder than JSP as it is html in java.	JSP is easy to code as it is java in html.														
Servlet plays a controller role in MVC approach.	JSP is the view in MVC approach for showing output.														
Servlet is faster than JSP.	JSP is slower than Servlet because the first step in JSP lifecycle is the translation of JSP to java code and then compile.														
Servlet can accept all protocol requests.	JSP only accept http requests.														
In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.														

	<p>In Servlet by default session management is not enabled, user have to enable it explicitly.</p>	<p>In JSP session management is automatically enabled.</p>			
	<p>In Servlet we have to implement everything like business logic and presentation logic in just one servlet file.</p>	<p>In JSP business logic is separated from presentation logic by using javaBeans.</p>			
	<p>Modification in Servlet is a time consuming task because it includes reloading, recompiling and restarting the server.</p>	<p>JSP modification is fast, just need to click the refresh button.</p>			
20.	<p>Discuss the need to use JSTL tags?</p> <h3>III. JSTL (JSP Standard Tag Library)</h3> <p>The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.</p> <p>A. Advantage of JSTL</p> <ol style="list-style-type: none"> Fast Development JSTL provides many tags that simplify the JSP. Code Reusability We can use the JSTL tags on various pages. No need to use scriptlet tag It avoids the use of scriptlet tag. <p>B. JSTL Tags</p> <p>There JSTL mainly provides five types of tags:</p> <table border="1"> <thead> <tr> <th>Tag Name</th><th>Description</th></tr> </thead> </table>			Tag Name	Description
Tag Name	Description				

	<u>Core tags</u>	The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is http://java.sun.com/jsp/jstl/core . The prefix of core tag is c .
	<u>Function tags</u>	The functions tags provide support for string manipulation and string length. The URL for the functions tags is http://java.sun.com/jsp/jstl/functions and prefix is fn .
	<u>Formatting tags</u>	The Formatting tags provide support for message formatting, number date formatting, etc. The URL for the Formatting tags is http://java.sun.com/jsp/jstl/fmt and prefix is fmt .
	<u>XML tags</u>	The XML tags provide flow control, transformation, etc. The URL for the XML tags is http://java.sun.com/jsp/jstl/xml and prefix is x .
	<u>SQL tags</u>	The JSTL SQL tags provide SQL support. The URL for the SQL tags is http://java.sun.com/jsp/jstl/sql and prefix is sql .

PART-B

Q.No	Questions
1.	(i) Integrate how servlets work and its life cycle. (ii) Explain and develop the Servlet API.
2.	(i) Analyze a JavaScript to find factorial of a given number. <pre>function factorial(x) { if (x === 0) { return 1; } return x * factorial(x-1); } console.log(factorial(5));</pre> <p>OUTPUT : 120</p>

(ii) **Differentiate** GET and POST method.

Get vs. Post

There are many differences between the Get and Post request. Let's see these differences:

GET	POST
1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
2) Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.
3) Get request can be bookmarked.	Post request cannot be bookmarked.
4) Get request is idempotent . It means second request will be ignored until response of first request is delivered	Post request is non-idempotent .
5) Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.

GET and POST

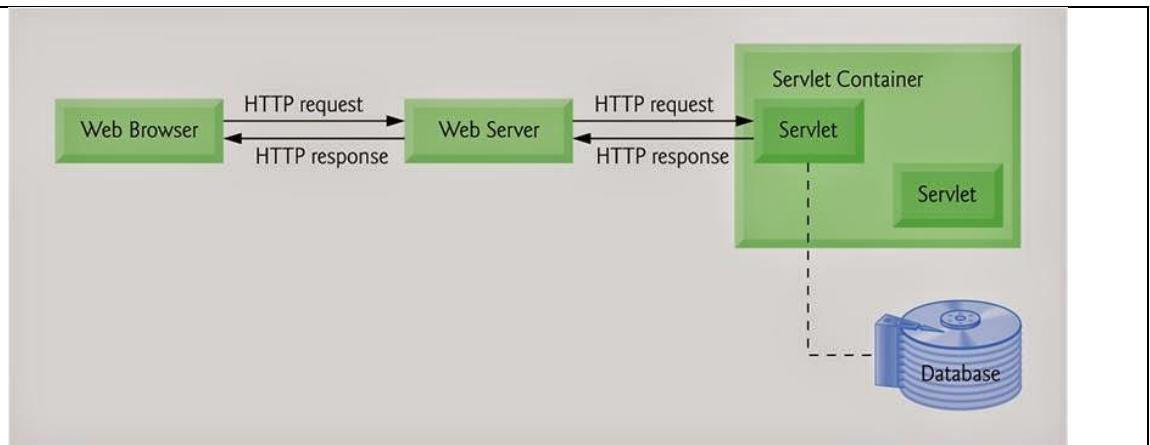
Two common methods for the request-response between a server and client are:

- o **GET**- It requests the data from a specified resource
- o **POST**- It submits the processed data to a specified resource

Demonstrate the Servlet architecture and explain its working principle.

3.

Servlet Architecture



Servlets read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.

Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.

Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.

Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.

Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Servlet API:

Servlet API contains three packages

javax.servlet: Package contains a number of classes and interfaces that describe the contract

between a servlet class and the runtime environment provided for an instance of such a class a conforming servelt container.

javax.servlet.annotation: Package contains a number of annotations that allow users to use annotations to declare servlets , filters, listeners and specify the metadata for the declared component

javax.servlet.http: Package contains a number of classes and interfaces that describe and define the contract between a servlet class rnning under the HTTP protocol and the runtime environment provided for an instance of such class by a confirming servlet container.

4.

Consider a database that has a table Employee with two columns Employee Id and Name. Assume that the administrator user id and password to access to access the database table are Scott and Tiger. Write a JDBC program that can query and print all entries in the

	<p>table employee. Make the database using type 2 driver database.driver and connection string jdbc :db.oci.</p>
	<p>Describe in detail the session handling in server side programming.</p> <h4>IV. Managing Session in Servlets</h4> <p>We all know that HTTP is a stateless protocol. All requests and responses are independent. But sometimes you need to keep track of client's activity across multiple requests. For eg. When a User logs into your website, not matter on which web page he visits after logging in, his credentials will be with the server, until he logs out. So this is managed by creating a session.</p> <p>Session Management is a mechanism used by the Web container to store session information for a particular user. There are four different techniques used by Servlet application for session management. They are as follows:</p> <ol style="list-style-type: none"> 1. Cookies 2. Hidden form field 3. URL Rewriting 4. HttpSession <p>Session is used to store everything that we can get from the client from all the requests the client makes.</p>
5.	<p>A. How Session Works</p> <pre> graph LR Client1[Client 1] -- "request id=123" --> WebContainer[Web Container] Client2[Client 2] -- "request id=134" --> WebContainer subgraph WebContainer [Web Container] servlet[servlet] S1[Session id=123] S2[Session id=134] Client1 --- servlet Client2 --- servlet servlet --- S1 servlet --- S2 end </pre> <p>B.</p> <p>The basic concept behind session is, whenever a user starts using our application, we can save a unique identification information about him, in an object which is available throughout the application, until its destroyed. So wherever the user goes, we will always have his information and we can always manage which user is doing what. Whenever a user wants to exit from your application, destroy the object with his information.</p>

	<p>(i) Discuss about JSTL.</p> <h2>V. JSTL (JSP Standard Tag Library)</h2> <p>The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.</p> <h3>A. Advantage of JSTL</h3> <ol style="list-style-type: none"> Fast Development JSTL provides many tags that simplify the JSP. Code Reusability We can use the JSTL tags on various pages. No need to use scriptlet tag It avoids the use of scriptlet tag. <h3>B. JSTL Tags</h3> <p>There JSTL mainly provides five types of tags:</p> <table border="1"> <thead> <tr> <th>Tag Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><u>Core tags</u></td> <td>The JSTL core tag provide variable support, URL management, flow etc. The URL for the core tag is http://java.sun.com/jsp/jstl/core. The prefix of core tag is c.</td> </tr> <tr> <td><u>Function tags</u></td> <td>The functions tags provide support for string manipulation and string conversion. The URL for the functions tags is http://java.sun.com/jsp/jstl/functions and prefix is fn.</td> </tr> <tr> <td><u>Formatting tags</u></td> <td>The Formatting tags provide support for message formatting, number formatting, date formatting, etc. The URL for the Formatting tags is http://java.sun.com/jsp/jstl/fmt and prefix is fmt.</td> </tr> <tr> <td><u>XML tags</u></td> <td>The XML tags provide flow control, transformation, etc. The URL for the XML tags is http://java.sun.com/jsp/jstl/xml and prefix is x.</td> </tr> <tr> <td><u>SQL tags</u></td> <td>The JSTL SQL tags provide SQL support. The URL for the SQL tags is http://java.sun.com/jsp/jstl/sql and prefix is sql.</td> </tr> </tbody> </table> <p>(ii) Summarize a client server JSP program to find simple interest and display the result in client.</p>	Tag Name	Description	<u>Core tags</u>	The JSTL core tag provide variable support, URL management, flow etc. The URL for the core tag is http://java.sun.com/jsp/jstl/core . The prefix of core tag is c .	<u>Function tags</u>	The functions tags provide support for string manipulation and string conversion. The URL for the functions tags is http://java.sun.com/jsp/jstl/functions and prefix is fn .	<u>Formatting tags</u>	The Formatting tags provide support for message formatting, number formatting, date formatting, etc. The URL for the Formatting tags is http://java.sun.com/jsp/jstl/fmt and prefix is fmt .	<u>XML tags</u>	The XML tags provide flow control, transformation, etc. The URL for the XML tags is http://java.sun.com/jsp/jstl/xml and prefix is x .	<u>SQL tags</u>	The JSTL SQL tags provide SQL support. The URL for the SQL tags is http://java.sun.com/jsp/jstl/sql and prefix is sql .
Tag Name	Description												
<u>Core tags</u>	The JSTL core tag provide variable support, URL management, flow etc. The URL for the core tag is http://java.sun.com/jsp/jstl/core . The prefix of core tag is c .												
<u>Function tags</u>	The functions tags provide support for string manipulation and string conversion. The URL for the functions tags is http://java.sun.com/jsp/jstl/functions and prefix is fn .												
<u>Formatting tags</u>	The Formatting tags provide support for message formatting, number formatting, date formatting, etc. The URL for the Formatting tags is http://java.sun.com/jsp/jstl/fmt and prefix is fmt .												
<u>XML tags</u>	The XML tags provide flow control, transformation, etc. The URL for the XML tags is http://java.sun.com/jsp/jstl/xml and prefix is x .												
<u>SQL tags</u>	The JSTL SQL tags provide SQL support. The URL for the SQL tags is http://java.sun.com/jsp/jstl/sql and prefix is sql .												

7.	<p>Explain the use of cookies for tracking for tracking requests with a program.</p> <h2>Session Tracking in JSP</h2> <p>Session Tracking :</p> <p>HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a new connection to the Web server and the server does not keep any record of previous client request.Session tracking is a mechanism that is used to maintain state about a series of requests from the same user(requests originating from the same browser) across some period of time. A session id is a unique token number assigned to a specific user for the duration of that user's session.</p> <p>Need Of Session Tracking :</p> <p>HTTP is a stateless protocol so When there is a series of continuous request and response from a same client to a server, the server cannot identify which client is sending request.If we want to maintain the conversational state,session tracking is needed. For example, in a shopping cart application a client keeps on adding items into his cart using multiple requests.When every request is made,the server should identify in which client's cart the item is to be added. So in this scenario, there is a certain need for session tracking.</p> <p>Solution is, when a client makes a request it should introduce itself by providing unique identifier every time.There are four ways to maintain session between web client and web server.</p> <p>Methods to track session :</p> <ul style="list-style-type: none"> Cookies URL Rewriting Hidden Fields Session API <p>Cookies :</p> <p>Cookies mostly used for session tracking. Cookie is a key value pair of information, sent by the server to the browser. This should be saved by the browser in its space in the client computer.</p>

	<p>Whenever the browser sends a request to that server it sends the cookie along with it. Then the server can identify the client using the cookie.</p> <p>This is not an effective way because many time browser does not support a cookie or users can opt to disable cookies using their browser preferences. In such case, the browser will not save the cookie at client computer and session tracking fails.</p>
8.	<p>(i) Explain about the standard actions in JSP.</p> <p>Actions are used for controlling the behavior of servlet engine.</p> <p>How many standard Action Tags are available in JSP?</p> <p>There are 11 types of Standard Action Tags as following:</p> <ul style="list-style-type: none"> • jsp:useBean • jsp:include • jsp:setProperty • jsp:getProperty • jsp:forward • jsp:plugin • jsp:attribute • jsp:body • jsp:text • jsp:param • jsp:attribute • jsp:output <p>(ii) Analyze MVC architecture of JSP.</p> <h2>VI. MVC in JSP</h2> <ol style="list-style-type: none"> 1. <u>MVC in JSP</u> 2. <u>Example of following MVC in JSP</u> <p>MVC stands for Model View and Controller. It is a design pattern that separates the business logic, presentation logic and data.</p> <p>Controller acts as an interface between View and Model. Controller intercepts all the incoming requests.</p> <p>Model represents the state of the application i.e. data. It can also have business logic.</p>

	<p>View represents the presentation i.e. UI(User Interface).</p> <p>a) Advantage of MVC (Model 2) Architecture</p> <ol style="list-style-type: none"> 1. Navigation Control is centralized 2. Easy to maintain the large application <pre> graph LR View[View (JSP)] --> Model[Model (Java Bean)] Model --> Database[Database] Controller[Controller (Filter/Servlet)] --> View Controller --> Model View --> Controller Database --> Model </pre> <p>The diagram illustrates the MVC (Model 2) architecture. It features a 'Container' boundary defined by a dashed line. Inside this container, there is a 'Controller (Filter/Servlet)' box at the top. Below it are two boxes: 'View (JSP)' on the left and 'Model (Java Bean)' on the right. Arrows point from the View to the Model, and from the Model to the Database. Outside the Container, a monitor icon represents the User Interface (View). An arrow points from the View to the Controller. Another arrow points from the Controller to the Model. A final arrow points from the Database to the Model.</p>
9.	<p>Explain in detail about Servlet database connectivity with an example of student database.</p> <h2>Example of Registration form in servlet</h2> <p>Table creation :</p> <pre> CREATE TABLE "REGISTERUSER" ("NAME" VARCHAR2(4000), "PASS" VARCHAR2(4000), "EMAIL" VARCHAR2(4000), "COUNTRY" VARCHAR2(4000)) /</pre> <h2>Example of Registration form in servlet</h2> <p>In this example, we have created the three pages.</p> <ul style="list-style-type: none"> o register.html o Register.java

- o web.xml

register.html

In this page, we have getting input from the user using text fields and combobox. The information entered by the user is forwarded to Register servlet, which is responsible to store the data into the database.

```
<html>
<body>
<form action="servlet/Register" method="post">

Name:<input type="text" name="userName"/><br/><br/>
Password:<input type="password" name="userPass"/><br/><br/>
Email Id:<input type="text" name="userEmail"/><br/><br/>
Country:
<select name="userCountry">
. <option>India</option>
. <option>Pakistan</option>
. <option>other</option>
. </select>
.
. <br/><br/>
. <input type="submit" value="register"/>
.
. </form>
. </body>
. </html>
```

Register.java

This servlet class receives all the data entered by user and stores it into the database. Here, we are performing the database logic. But you may separate it, which will be better for the web application.

```
import java.io.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class Register extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
}
```

```

. PrintWriter out = response.getWriter();
.

. String n=request.getParameter("userName");
. String p=request.getParameter("userPass");
. String e=request.getParameter("userEmail");
. String c=request.getParameter("userCountry");
.

. try{
.   Class.forName("oracle.jdbc.driver.OracleDriver");
.   Connection con=DriverManager.getConnection(
.     "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
.

.   PreparedStatement ps=con.prepareStatement(
.     "insert into registeruser values(?, ?, ?, ?)");
.

.   ps.setString(1,n);
.   ps.setString(2,p);
.   ps.setString(3,e);
.   ps.setString(4,c);
.

.   int i=ps.executeUpdate();
.   if(i>0)
.     out.print("You are successfully registered...");
.

.

. }catch (Exception e2) {System.out.println(e2);}
.

. out.close();
.
.
.
```

10.	<p>Demonstrate the procedure of installing and configuring Apache Tomcat.</p> <h2>How To Install Apache Tomcat 8 on Ubuntu 16.04</h2> <p>Apache Tomcat is a web server and servlet container that is used to serve Java applications. Tomcat is an open source implementation of the Java Servlet and JavaServer Pages technologies, released by the Apache Software Foundation. This</p>
-----	--

tutorial covers the basic installation and some configuration of the latest release of Tomcat 8 on your Ubuntu 16.04 server.

Step 1: Install Java

Tomcat requires Java to be installed on the server so that any Java web application code can be executed. We can satisfy that requirement by installing OpenJDK with apt-get.

First, update your apt-get package index:

```
sudo apt-get update
```

Then install the Java Development Kit package with apt-get:

```
sudo apt-get install default-jdk
```

Now that Java is installed, we can create a `tomcat` user, which will be used to run the Tomcat service.

Step 2: Create Tomcat User

For security purposes, Tomcat should be run as an unprivileged user (i.e. not root). We will create a new user and group that will run the Tomcat service.

First, create a new `tomcat` group:

```
sudo groupadd tomcat
```

Next, create a new `tomcat` user. We'll make this user a member of the `tomcat` group, with a home directory of `/opt/tomcat` (where we will install Tomcat), and with a shell of `/bin/false` (so nobody can log into the account):

```
sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

Now that our `tomcat` user is set up, let's download and install Tomcat.

Step 3: Install Tomcat

The best way to install Tomcat 8 is to download the latest binary release then configure it manually.

We will install Tomcat to the /opt/tomcat directory. Create the directory, then extract the archive to it with these commands:

```
sudo mkdir /opt/tomcat
```

```
sudo tar xzvf apache-tomcat-8*tar.gz -C /opt/tomcat --strip-components=1
```

Next, we can set up the proper user permissions for our installation.

Step 4: Update Permissions

The tomcat user that we set up needs to have access to the Tomcat installation. We'll set that up now.

Change to the directory where we unpacked the Tomcat installation:

```
cd /opt/tomcat
```

Give the tomcat group ownership over the entire installation directory:

```
sudo chgrp -R tomcat /opt/tomcat
```

Next, give the tomcat group read access to the conf directory and all of its contents, and **execute** access to the directory itself:

```
sudo chmod -R g+r conf
```

```
sudo chmod g+x conf
```

Make the tomcat user the owner of the webapps, work, temp, and logs directories:

```
sudo chown -R tomcat webapps/ work/ temp/ logs/
```

Now that the proper permissions are set up, we can create a systemd service file to manage the Tomcat process.

Step 5: Create a systemd Service File

We want to be able to run Tomcat as a service, so we will set up systemd service file.

Tomcat needs to know where Java is installed. This path is commonly referred to as “JAVA_HOME”. The easiest way to look up that location is by running this command:

Step 6: Adjust the Firewall and Test the Tomcat Server

Now that the Tomcat service is started, we can test to make sure the default page is available.

Before we do that, we need to adjust the firewall to allow our requests to get to the service. If you followed the prerequisites, you will have a ufw firewall enabled currently.

Tomcat uses port 8080 to accept conventional requests. Allow traffic to that port by typing:

```
sudo ufw allow 8080
```

Step 7: Configure Tomcat Web Management Interface

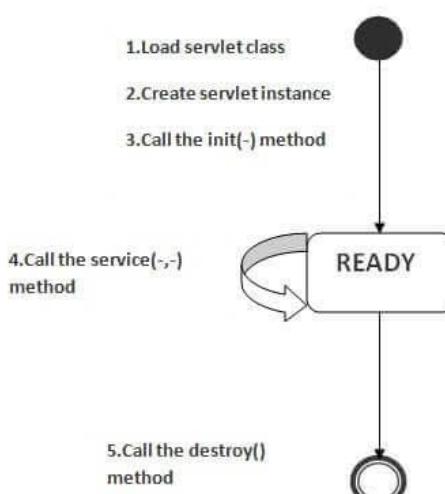
Step 8: Access the Web Interface

Now that we have created a user, we can access the web management interface again in a web browser. Once again, you can get to the correct interface by entering your server’s domain name or IP address followed on port 8080 in your browser:

Open in web browser

`http://server_domain_or_IP:8080`

The page you see should be the same one you were given when you tested earlier:

	<p>Home Documentation Configuration Examples Wiki Mailing Lists Find Help</p> <p>Apache Tomcat/8.0.33</p> <p>If you're seeing this, you've successfully installed Tomcat. Congratulations!</p> <p>The Apache Software Foundation http://www.apache.org/</p> <p> Recommended Reading: Security Considerations HOW-TO Manager Application HOW-TO Clustering/Session Replication HOW-TO</p> <p>Server Status Manager App Host Manager</p> <p>Developer Quick Start</p> <table border="0"> <tr> <td>Tomcat Setup</td><td>Realms & AAA</td><td>Examples</td><td>Servlet Specifications</td></tr> <tr> <td>First Web Application</td><td>JDBC DataSources</td><td></td><td>Tomcat Versions</td></tr> </table> <p>Your installation of Tomcat is complete! You are now free to deploy your own Java web applications!</p>	Tomcat Setup	Realms & AAA	Examples	Servlet Specifications	First Web Application	JDBC DataSources		Tomcat Versions
Tomcat Setup	Realms & AAA	Examples	Servlet Specifications						
First Web Application	JDBC DataSources		Tomcat Versions						
11.	<p>(i) Discuss about the Servlet life cycle.</p> <h2>VII. Life Cycle of a Servlet (Servlet Life Cycle)</h2> <p>The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:</p> <ol style="list-style-type: none"> 1. Servlet class is loaded. 2. Servlet instance is created. 3. init method is invoked. 4. service method is invoked. 5. destroy method is invoked.  <pre> graph TD A(()) -- "1.Load servlet class" --> B[Create servlet instance] B --> C(()) C -- "3.Call the init(-) method" --> D[READY] D -- "4.Call the service(-,-) method" --> E((())) E -- "5.Call the destroy() method" --> F((()) </pre> <p>The diagram illustrates the five steps of the servlet life cycle. It starts with a black circle (Step 1), followed by a rounded rectangle labeled 'CREATE' (Step 2). This leads to a black circle containing the word 'READY' (Step 3). From the 'READY' state, an arrow points to a rounded rectangle labeled 'SERVICE' (Step 4). Finally, an arrow points from 'SERVICE' to an empty black circle (Step 5).</p>								

	<p>As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.</p> <p>(ii) List JSP advantages.</p> <ol style="list-style-type: none"> 1. Advantages of JSP over Servlet <p>There are many advantages of JSP over the Servlet. They are as follows:</p> <ol style="list-style-type: none"> a) 1) Extension to Servlet JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy. b) 2) Easy to maintain JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic. c) 3) Fast Development: No need to recompile and redeploy If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application. d) 4) Less code than Servlet In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.
12.	<p>(i) Explain and write a simple JDBC program.</p> <h3>Java Database Connectivity with MySQL</h3> <p>To connect Java application with the MySQL database, we need to follow 5 following steps.</p> <ol style="list-style-type: none"> 1. Driver class: The driver class for the mysql database is com.mysql.jdbc.Driver. 2. Connection URL: The connection URL for the mysql database is jdbc:mysql://localhost:3306/sonoo where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we

may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.

3. **Username:** The default username for the mysql database is **root**.
4. **Password:** It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

```
create database sonoo;
use sonoo;
create table emp(id int(10),name varchar(40),age int(3));
```

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password both.

```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sonoo","root","root");
//here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
con.close();
} catch(Exception e){ System.out.println(e);}
}
}
```

(ii) **List** various JSP scripting components.

VIII. JSP Scripting Element

JSP Scripting element are written inside `<% %>` tags. These code inside `<% %>` tags are processed by the JSP engine during translation of the JSP page. Any other text in the JSP page is considered as HTML code or plain text.

Example:

```
html>
<head>
```

```

<title>My First JSP Page</title>
</head>
<%
    int count = 0;
%>
<body>
    Page Count is <% out.println(++count); %>
</body>
</html>

```

1. Types of scripting elements

Scripting Element	Example
Comment	<%-- comment --%>
Directive	<%@ directive %>
Declaration	<%! declarations %>
Scriptlet	<% scriptlets %>
Expression	<%= expression %>

B. JSP Comment

JSP Comment is used when you are creating a JSP page and want to put in comments about what you are doing. JSP comments are only seen in the JSP page. These comments are not included in servlet source code during translation phase, nor they appear in the HTTP response. Syntax of JSP comment is as follows :

```
<%-- JSP comment --%>
```

Simple Example of JSP Comment

```

<html>
    <head>
        <title>My First JSP Page</title>
    </head>
    <%
        int count = 0;
    %>
    <body>
        <%-- Code to show page count --%>
        Page Count is <% out.println(++count); %>
    </body>
</html>

```

13.

(i) **Demonstrate** with suitable example for core and formatting tags in JSTL.

JSTL Formatting tags

The formatting tags provide support for message formatting, number and date formatting etc. The url for the formatting tags is <http://java.sun.com/jsp/jstl/fmt> and prefix is **fmt**.

The JSTL formatting tags are used for internationalized web sites to display and format text, the time, the date and numbers. The syntax used for including JSTL formatting library in your JSP is:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
```

Formatting Tags	Descriptions
fmt:parseNumber	It is used to Parses the string representation of a currency, percentage or number.
fmt:timeZone	It specifies a parsing action nested in its body or the time zone for any time formatting.
fmt:formatNumber	It is used to format the numerical value with specific format or precision.
fmt:parseDate	It parses the string representation of a time and date.
fmt:bundle	It is used for creating the ResourceBundle objects which will be used by their tag body.
fmt:setTimeZone	It stores the time zone inside a time zone configuration variable.
fmt:setBundle	It loads the resource bundle and stores it in a bundle configuration variable or the named scoped variable.
fmt:message	It display an internationalized message.
fmt:formatDate	It formats the time and/or date using the supplied pattern and styles.

JSTL Core <c:choose>, <c:when>, <c:otherwise> Tag

The `<c:choose>` tag is a conditional tag that establish a context for mutually exclusive conditional operations. It works like a Java **switch** statement in which we choose between a numbers of alternatives.

Example

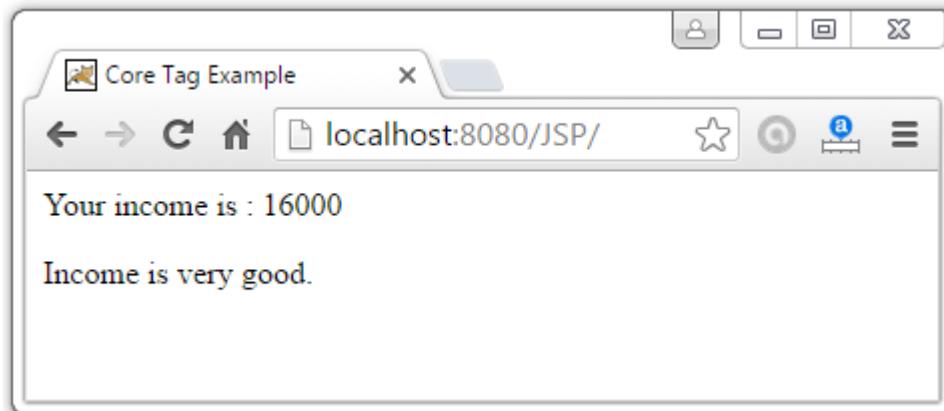
Let's see the simple example of < c:choose >, < c:when > < c:otherwise > tag:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${4000*4}" />
<p>Your income is : <c:out value="${income}" /></p>
<c:choose>
    <c:when test="${income <= 1000}">
        Income is not good.
    </c:when>
    <c:when test="${income > 10000}">
        Income is very good.
    </c:when>
    <c:otherwise>
        Income is undetermined...
    </c:otherwise>
</c:choose>
</body>
</html>
```

This will produce the following result:

Your income is : 16000

Income is very good.



(ii) Demonstrate with suitable example for SQL and XML tags in JSTL.

JSTL SQL

The **<sql:setDataSource>** tag sets the data source configuration variable or saves the data-source information in a scoped variable that can be used as input to the other JSTL database actions.

Attribute

The **<sql:setDataSource>** tag has the following attributes –

Attribute	Description
driver	Name of the JDBC driver class to be registered
url	JDBC URL for the database connection
user	Database username
password	Database password
password	Database password
dataSource	Database prepared in advance
var	Name of the variable to represent the database
scope	Scope of the variable to represent the database

Example

Consider the following information about your MySQL database setup –

- We are using **JDBC MySQL** driver.
- We are going to connect to TEST database on local machine.
- We would use **user_id** and **mypassword** to access TEST database.

All the above parameters will vary based on your MySQL or any other database setup. Considering the above parameters, following example uses the **setDataSource** tag –

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>  
<%@ taglib uri = "http://java.sun.com/jsp/jstl/sql" prefix = "sql"%>
```

```

<html>
  <head>
    <title>JSTL sql:setDataSource Tag</title>
  </head>

  <body>
    <sql:setDataSource var = "snapshot" driver = "com.mysql.jdbc.Driver"
      url = "jdbc:mysql://localhost/TEST"
      user = "user_id" password = "mypassword"/>
    <sql:query dataSource = "${snapshot}" sql = "..." var = "result" />
  </body>
</html>

```

JSTL XML

<x:parse> Tag

The <x:parse> tag is used for parse the XML data specified either in the tag body or an attribute. It is used for parse the xml content and the result will stored inside specified variable.

The syntax used for including the <x:parse> tag is:

<x:parse attributes> body content </x:parse>

EXAMPLE

Let us put the following content in **novels.xml** file:

```

<books>
  <book>
    <name>Three mistakes of my life</name>
    <author>Chetan Bhagat</author>
    <price>200</price>
  </book>
  <book>
    <name>Tomorrow land</name>
    <author>NUHA</author>
    <price>2000</price>

```

	<pre></book> </books></pre> <p>index.jsp</p> <p>(IN the same directory)</p> <pre><%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %> <html> <head> <title>x:parse Tag</title> </head> <body> <h2>Books Info:</h2> <c:import var="bookInfo" url="novels.xml"/> <x:parse xml="\${bookInfo}" var="output"/> <p>First Book title: <x:out select="\$output/books/book[1]/name" /></p> <p>First Book price: <x:out select="\$output/books/book[1]/price" /></p> <p>Second Book title: <x:out select="\$output/books/book[2]/name" /></p> <p>Second Book price: <x:out select="\$output/books/book[2]/price" /></p> </body> </html></pre> <p>Output:</p> <p>Books Info:</p> <p>First Book title: Three mistakes of my life</p> <p>First Book price: 200</p> <p>Second Book title: Tomorrow land</p> <p>Second Book price: 2000</p>
14.	<p>Define HTML and JSP. Use the same and design a scientific calculator.</p> <p>Calculator.jsp</p> <pre><%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> <html> <head> <title>Calculator</title></pre>

```

<style>
    h1 {
        font-family: Arial;
        font-size: 14pt;
        font-weight: normal;
    }
    td input {
        font-family: Arial;
        font-size: 10pt;
        width: 30px;
    }

    input.double {
        width: 60px;
    }

    input.doubleheight {
        height: 48px;
    }

    input.display {
        border: 1px solid black;
        readonly: true;
        width: 120px;
        padding: 2px;
    }
</style>
</head>

<body>
    <h1>Calculator</h1>

    <form method="GET" action="calculate.html">
        <table border="0" cellpadding="0" cellspacing="0">
            <tr>
                <td colspan="4">
                    <input class="display" type="text" id="display" value=<c:out
value="${displayAmount}" /> readonly/>
                </td>
            </tr>
            <tr>
                <td><input type="submit" name="button" id="btn-7"
value="7"/></td>
                <td><input type="submit" name="button" id="btn-8"
value="8"/></td>
                <td><input type="submit" name="button" id="btn-9"
value="9"/></td>
                <td><input type="submit" name="button" id="btn-/"
value="/" /></td>
            </tr>
        </table>
    </form>
</body>

```

```

<td><input type="submit" name="button" id="btn-C"
value="C"/></td>
</tr>
<tr>
<td><input type="submit" name="button" id="btn-4"
value="4"/></td>
<td><input type="submit" name="button" id="btn-5"
value="5"/></td>
<td><input type="submit" name="button" id="btn-6"
value="6"/></td>
<td><input type="submit" name="button" id="btn-*"
value="*"/></td>
<td></td>
</tr>
<tr>
<td><input type="submit" name="button" id="btn-1"
value="1"/></td>
<td><input type="submit" name="button" id="btn-2"
value="2"/></td>
<td><input type="submit" name="button" id="btn-3"
value="3"/></td>
<td><input type="submit" name="button" id="btn--" value="-"
"/></td>
<td rowspan="2"><input class="doubleheight" type="submit"
name="button" id="btn-=" value="/" /></td>
</tr>
<tr>
<td colspan="2"><input class="double" type="submit"
name="button" id="btn-0" value="0" /></td>
<td><input type="button" name="button" id="btn-."
value"." /></td>
<td><input type="submit" name="button" id="btn-+"
value "+" /></td>
</tr>
</table>
</form>
</body>
</html>

```

PART – C

Q.No	Questions
1.	Design a HTML forms by embedding JSP code for submission of a resume to a job portal website with appropriate database connectivity.
2.	Evaluate a complete query application for books database using JDBC.
3.	Write a program that allows the user to select a favourite programming language and post the choice to the server. The response is a web page in which the user can click a

	link to view a list of book recommendations. The cookies previously stored on the client are read by the servlet and form a web page containing the book recommendation. Use servlet cookies and HTML.
4.	Develop a JSP program to display the grade of a student by accepting the marks of five subjects.

CS8651 Internet Programming – 2017Reg

PHP: An introduction to PHP – Using PHP – Variables – Program control – Built-in functions – Form Validation – Regular Expressions – File handling – Cookies – Connecting to Database; **XML:** Basic XML – Document Type Definition – XML Schema DOM and Presenting XML, XML Parsers and Validation, XSL and XSLT Transformation, News Feed (RSS and ATOM).

Internet Programming – UNIT-IV

Q.No	Questions
	<p>Define PHP. List the features.</p> <p>What is PHP?</p> <ul style="list-style-type: none">• PHP is an acronym for "PHP: Hypertext Preprocessor"• PHP is a widely-used, open source scripting language• PHP scripts are executed on the server• PHP is free to download and use <p>PHP is an amazing and popular language!</p> <p>It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)! It is deep enough to run the largest social network (Facebook)! It is also easy enough to be a beginner's first server side language!</p> <p>What is a PHP File?</p> <p>1.</p> <ul style="list-style-type: none">• PHP files can contain text, HTML, CSS, JavaScript, and PHP code• PHP code is executed on the server, and the result is returned to the browser as plain HTML• PHP files have extension ".php" <p>What Can PHP Do?</p> <ul style="list-style-type: none">• PHP can generate dynamic page content• PHP can create, open, read, write, delete, and close files on the server• PHP can collect form data• PHP can send and receive cookies• PHP can add, delete, modify data in your database• PHP can be used to control user-access• PHP can encrypt data <p>With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.</p>

List the rules for creating variables in PHP.

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

2.

- A variable starts with the \$ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (\$age and \$AGE are two different variables)
- PHP variable names are case-sensitive

Illustrate a PHP program to determine the type of browser that a web client is using.

Display the Browser – PHP Script

The following PHP function can be used to display the browser:

<?php

```
function get_the_browser()
{
    if(strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== false)
        return 'Internet explorer';
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Trident') !== false)
        return 'Internet explorer';
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Firefox') !== false)
        return 'Mozilla Firefox';
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Chrome') !== false)
        return 'Google Chrome';
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Opera Mini') !== false)
        return "Opera Mini";
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Opera') !== false)
        return "Opera";
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Safari') !== false)
        return "Safari";
    else
        return 'Other';
```

3.

```
}
```

```
?>
```

In the above code, we are checking each possible browser that may be and return the browser name. Here we haven't checked the Mozilla because of most of the browser using this as the user agent string.

Below is how to display the browser name on our web page:

```
Echo get_the_browser();
```

Name any four built-in functions in PHP.

PHP Reference

The PHP reference contains different categories of all PHP functions and constants, along with examples.



Infer when should the super global arrays in PHP be used?

Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes.

PHP Global Variables - Superglobals

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- `$GLOBALS`
- `$_SERVER`
- `$_REQUEST`
- `$_POST`
- `$_GET`
- `$_FILES`
- `$_ENV`
- `$_COOKIE`
- `$_SESSION`

Which super global array in PHP would contain a HTML form's POST data?

PHP Superglobal - `$_POST`

Super global variables are built-in variables that are always available in all scopes.

PHP \$ POST

PHP `$_POST` is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". `$_POST` is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the `<form>` tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable `$_POST` to collect the value of the input field:

Example

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
    Name: <input type="text" name="fname">
    <input type="submit">
</form>
```

```

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>

```

Classify the difference between echo() & print() functions.

PHP echo and print Statements

With PHP, there are two basic ways to get output: **echo** and **print**.

echo and **print** are more or less the same. They are both used to output data to the screen.

The differences are small:

The PHP echo Statement	The PHP print Statement
<ul style="list-style-type: none"> ➤ echo has no return value ➤ echo can take multiple parameters ➤ echo is marginally faster than print. 	<ul style="list-style-type: none"> ➤ print has a return value of 1 so it can be used in expressions. ➤ print can take one argument
The echo statement can be used with or without parentheses: echo or echo() .	The print statement can be used with or without parentheses: print or print() .
Example-1 <?php echo "<h2>PHP is Fun!</h2>"; echo "Hello world!
"; echo "I'm about to learn PHP!
"; echo "This ", "string ", "was ", "made ", "with multiple parameters. "; ?>	Example-1 <?php print "<h2>PHP is Fun!</h2>"; print "Hello world!
"; print "I'm about to learn PHP!"; ?>
PHP is Fun! Hello world!	PHP is Fun! Hello world!

	<p>Example-2</p> <pre><?php \$txt1 = "Learn PHP"; \$txt2 = "W3Schools.com"; \$x = 5; \$y = 4; echo "<h2>" . \$txt1 . "</h2>"; echo "Study PHP at " . \$txt2 . "
"; echo \$x + \$y; ?></pre>	<p>Example-2</p> <pre><?php \$txt1 = "Learn PHP"; \$txt2 = "W3Schools.com"; \$x = 5; \$y = 4; print "<h2>" . \$txt1 . "</h2>"; print "Study PHP at " . \$txt2 . "
"; print \$x + \$y; ?></pre>									
	<p>Learn PHP Study PHP at W3Schools.com 9</p>	<p>Learn PHP Study PHP at W3Schools.com 9</p>									
7.	<p>List any two advantages of XML document.</p> <p>Using XML to exchange information offers many benefits.</p> <p>Advantages of XML include the following:</p> <ul style="list-style-type: none"> ➤ XML uses human, not computer, language. XML is readable and understandable, even by novices, and no more difficult to code than HTML. ➤ XML is completely compatible with Java™ and 100% portable. Any application that can process XML can use your information, regardless of platform. ➤ XML is extendable. Create your own tags, or use tags created by others, that use the natural language of your domain, that have the attributes you need, and that makes sense to you and your users. 										
8.	<p>Give the difference between DTD and XML schema for defining XML document structure with appropriate examples.</p> <h3>DTD vs XSD</h3> <p>There are many differences between DTD (Document Type Definition) and XSD (XML Schema Definition). In short, DTD provides less control on XML structure whereas XSD (XML schema) provides more control.</p> <p>The important differences are given below:</p> <table border="1"> <thead> <tr> <th>No.</th><th>DTD(Document Type Definition)</th><th>XSD(XML Schema Definition)</th></tr> </thead> <tbody> <tr> <td>1)</td><td>DTD stands for Document Type Definition.</td><td>XSD stands for XML Schema Definition.</td></tr> <tr> <td>2)</td><td>DTDs are derived from SGML syntax.</td><td>XSDs are written in XML.</td></tr> </tbody> </table>		No.	DTD (Document Type Definition)	XSD (XML Schema Definition)	1)	DTD stands for Document Type Definition.	XSD stands for XML Schema Definition.	2)	DTDs are derived from SGML syntax.	XSDs are written in XML.
No.	DTD (Document Type Definition)	XSD (XML Schema Definition)									
1)	DTD stands for Document Type Definition.	XSD stands for XML Schema Definition.									
2)	DTDs are derived from SGML syntax.	XSDs are written in XML.									

	3)	DTD doesn't support datatypes.	XSD supports datatypes for elements and attributes.	
	4)	DTD doesn't support namespace.	XSD supports namespace.	
	5)	DTD doesn't define order for child elements.	XSD defines order for child elements.	
	6)	DTD is not extensible.	XSD is extensible.	
	7)	DTD is not simple to learn.	XSD is simple to learn because you don't need to learn new language.	
	8)	DTD provides less control on XML structure.	XSD provides more control on XML structure.	
	Analyze about Query String in PHP.			
9.	<h2>Query string</h2> <p>The information can be sent across the web pages. This information is called query string. This query string can be passed from one page to another by appending it to the address of the page. You can pass more than one query string by inserting the & sign between the query strings. A query string can contain two things: the query string ID and its value. The query string passed across the web pages is stored in <code>\$_REQUEST</code>, <code>\$_GET</code>, or <code>\$_POST</code> variable.</p> <hr/> <p>Query string handling in PHP</p> <hr/> <p>Query strings</p> <hr/> <p>To access the data in a query string you can use the <code>\$_GET</code> global array. Each element in this array has a key which is the name of the query string variable and a value which is the value of that variable.</p> <hr/> <pre>my link</pre>			

This link loads the page mypage.php with two variables *variable1* and *variable2* with values value1 and value2 respectively.

```
echo $_GET['variable1'];
echo $_GET['variable2'];
// outputs:
//value1
//value2
```

Form data

The get method of forms sends the data to a page via a query string.

```
<form name="form1" id="form1" method="get" action="">
  <input name="textbox" id="textbox" type="text" value="value1" />
  <input name="textbox2" id="textbox2" type="text" value="value2" />
  <input type="submit" name="submitbutton" id="submitbutton" value="Submit" />
</form>
```

This form passes the value of the two text boxes to the page myform.php.

```
print_r($_GET);
// outputs:
// Array (
//   [textbox] => value1
//   [textbox2] => value2
//   [submitbutton] => Submit
// )

echo $_GET['textbox'];
//outputs: value1
```

Show an example for XML namespace.

10. A **Namespace** is a set of unique names. Namespace is a mechanisms by which element and attribute name can be assigned to a group. The Namespace is identified by URI(Uniform Resource Identifiers).

Namespace Declaration

A Namespace is declared using reserved attributes. Such an attribute name must either be **xmlns** or begin with **xmlns:** shown as below –

```
<element xmlns:name = "URL">
```

Syntax

- The Namespace starts with the keyword **xmlns**.
- The word **name** is the Namespace prefix.
- The **URL** is the Namespace identifier.

Example

Namespace affects only a limited area in the document. An element containing the declaration and all of its descendants are in the scope of the Namespace. Following is a simple example of XML Namespace –

```
<?xml version = "1.0" encoding = "UTF-8"?>
<cont:contact xmlns:cont = "www.tutorialspoint.com/profile">
    <cont:name>Tanmay Patil</cont:name>
    <cont:company>Tutorialspoint</cont:company>
    <cont:phone>(011) 123-4567</cont:phone>
</cont:contact>
```

Here, the Namespace prefix is **cont**, and the Namespace identifier (URI) as www.tutorialspoint.com/profile. This means, the element names and attribute names with the **cont** prefix (including the contact element), all belong to the www.tutorialspoint.com/profile namespace.

Define XML parse tree.

An XML document is always descriptive. The tree structure is often referred to as **XML Tree** and plays an important role to describe any XML document easily.

The tree structure contains root (parent) elements, child elements and so on. By using tree structure, you can get to know all succeeding branches and sub-branches starting from the root. The parsing starts at the root, then moves down the first branch to an element, take the first branch from there, and so on to the leaf nodes.

11.

Example

Following example demonstrates simple XML tree structure –

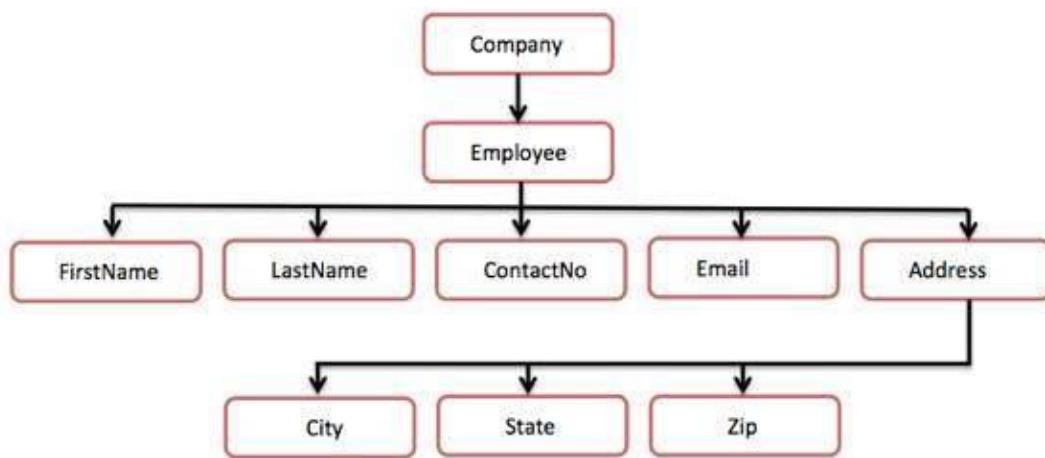
```
<?xml version = "1.0"?>
<Company>
    <Employee>
        <FirstName>Tanmay</FirstName>
```

```

<LastName>Patil</LastName>
<ContactNo>1234567890</ContactNo>
<Email>tanmaypatil@xyz.com</Email>
<Address>
    <City>Bangalore</City>
    <State>Karnataka</State>
    <Zip>560212</Zip>
</Address>
</Employee>
</Company>

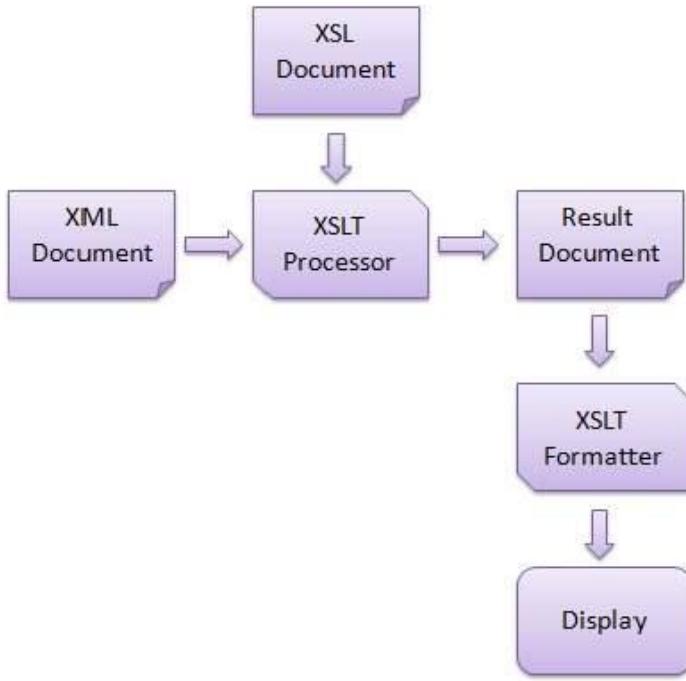
```

Following tree structure represents the above XML document –



In the above diagram, there is a root element named as <company>. Inside that, there is one more element <Employee>. Inside the employee element, there are five branches named <FirstName>, <LastName>, <ContactNo>, <Email>, and <Address>. Inside the <Address> element, there are three sub-branches, named <City> <State> and <Zip>.

12.	<p>Identify why XSLT is an important tool for development of web applications.</p> <h2>What is XSLT</h2> <p>XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.</p> <h3>How XSLT Works</h3> <p>An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.</p>
-----	---



Advantages

Here are the advantages of using XSLT –

- Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
- Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

Assess the data types in XML schema.

You can define XML schema elements in the following ways –

Simple Type

Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example –

```
<xs:element name = "phone_number" type = "xs:int" />
```

Complex Type

13. A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example –

```
<xs:element name = "Address">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string" />
      <xs:element name = "company" type = "xs:string" />
      <xs:element name = "phone" type = "xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

	<pre></xs:complexType> </xs:element></pre>
14.	<p>Explain DTD for XML Schemas.</p> <p>A document type definition (DTD) is a set of <i>markup declarations</i> that define a <i>document type</i> for a SGML-family markup language (GML, SGML, XML, HTML).</p> <p>A DTD defines the valid building blocks of an XML document. It defines the document structure with a list of validated elements and attributes. A DTD can be declared inline inside an XML document, or as an external reference.</p> <h3><u>XML DTD schema example</u></h3> <p>An example of a very simple external XML DTD to describe the schema of a list of persons might consist of:</p> <pre><!ELEMENT people_list (person)*> <!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)> <!ELEMENT name (#PCDATA)> <!ELEMENT birthdate (#PCDATA)> <!ELEMENT gender (#PCDATA)> <!ELEMENT socialsecuritynumber (#PCDATA)></pre> <p>Taking this line by line:</p> <ol style="list-style-type: none"> 1. <code>people_list</code> is a valid element name, and an instance of such an element contains any number of <code>person</code> elements. The <code>*</code> denotes there can be 0 or more <code>person</code> elements within the <code>people_list</code> element. 2. <code>person</code> is a valid element name, and an instance of such an element contains one element named <code>name</code>, followed by one named <code>birthdate</code> (optional), then <code>gender</code> (also optional) and <code>socialsecuritynumber</code> (also optional). The <code>?</code> indicates that an element is optional. The reference to the <code>name</code> element name has no <code>?</code>, so a <code>person</code> element <i>must</i> contain a <code>name</code> element. 3. <code>name</code> is a valid element name, and an instance of such an element contains "parsed character data" (<code>#PCDATA</code>). 4. <code>birthdate</code> is a valid element name, and an instance of such an element contains parsed character data. 5. <code>gender</code> is a valid element name, and an instance of such an element contains parsed character data. 6. <code>socialsecuritynumber</code> is a valid element name, and an instance of such an element contains parsed character data. <p>An example of an XML file that uses and conforms to this DTD follows. The DTD is referenced here as an external subset, via the SYSTEM specifier and a URI. It assumes that we can identify the DTD with the relative URI reference "example.dtd"; the</p>

"people_list" after "DOCTYPE" tells us that the root tags, or the first element defined in the DTD, is called "people_list":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

Evaluate the process of displaying XML document in browser.

Display an XML Document in a Web Browser

Displaying XML Using CSS

XML stands for Extensible Markup Language. It is a dynamic markup language. It is used to transform data from one form to another form.

An XML file can be displayed using two ways. These are as follows :-

1. Cascading Style Sheet
2. Extensible Stylesheet Language Transformation

Displaying XML file using CSS :

CSS can be used to display the contents of the XML document in a clear and precise manner. It gives the design and style to whole XML document.

- **Basic steps in defining a CSS style sheet for XML :**

For defining the style rules for the XML document, the following things should be done :-

1. Define the style rules for the text elements such as font-size, color, font-weight, etc.
2. Define each element either as a block, inline or list element, using the display property of CSS.
3. Identify the titles and bold them.

- **Linking XML with CSS :**

In order to display the XML file using CSS, link XML file with CSS. Below is the syntax for linking the XML file with CSS:

```
<?xml-stylesheet type="text/css" href="name_of_css_file.css"?>
```

- **Example 1.**

In this example, the XML file is created that contains the information about five books and displaying the XML file using CSS.

XML file :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="Rule.css"?>
<books>
```

```

<heading>Welcome To GeeksforGeeks </heading>
<book>
    <title>Title :- Web Programming</title>
    <author>Author :- Chrisbates</author>
    <publisher>Publisher :- Wiley</publisher>
    <edition>Edition :- 3</edition>
    <price> Price :- 300</price>
</book>
<book>
    <title>Title :- Internet world-wide-web</title>
    <author>Author :- Ditel</author>
    <publisher>Publisher :- Pearson</publisher>
    <edition>Edition :- 3</edition>
    <price>Price :- 400</price>
</book>

</books>

```

In the above example, Books.xml is linked with Rule.css which contains the corresponding style sheet rules.

CSS FILE :

```

books {
    color: white;
    background-color : gray;
    width: 100%;
}
heading {
    color: green;
    font-size : 40px;
    background-color : powderblue;
}
heading, title, author, publisher, edition, price {
    display : block;
}
title {
    font-size : 25px;
    font-weight : bold;
}

```

- **Output :**



XML namespaces are used for providing uniquely named [elements](#) and attributes in an [XML](#) document. They are defined in a [W3C recommendation](#).^{[1][2]} An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a [namespace](#), the ambiguity between identically named elements or attributes can be resolved.

How to get rid of name conflict?

1) By Using a Prefix

You can easily avoid the XML namespace by using a name prefix.

```
<h:table>
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Note: In this example, you will get no conflict because both the tables have specific names.

2) By Using xmlns Attribute

You can use xmlns attribute to define namespace with the following syntax:

```
<element xmlns:name = "URL">
```

Let's see the example:

```
<root>
<h:table xmlns:h="http://www.abc.com/TR/html4/">
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.xyz.com/furniture">
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
```

```
. <f:length>120</f:length>
. </f:table>
. </root>
```

In the above example, the `<table>` element defines a namespace and when a namespace is defined for an element, the child elements with the same prefixes are associated with the same namespace.

Analyze on ATOM in RSS.

What is Atom 1.0 ?

Atom is the name of an XML-based Web content and metadata syndication format, and an application-level protocol for publishing and editing Web resources belonging to periodically updated websites.

Atom is a relatively recent spec and is much more robust and feature-rich than RSS. For instance, where RSS requires descriptive fields such as title and link only in item breakdowns, Atom requires these things for both items and the full Feed.

All Atom Feeds must be well-formed [XML](#) documents, and are identified with the `application/atom+xml` media type.

Structure of an Atom 1.0 Feed

A Feed consists of some metadata, followed by any number of entries. Here is a basic structure of an Atom 1.0 Feed.

17.

```
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <title>...</title>
    <link>...</link>
    <updated>...</updated>

    <author>
        <name>...</name>
    </author>

    <id>...</id>

    <entry>
        <title>...</title>
        <link>...</link>
        <id>...</id>

        <updated>...</updated>
        <summary>...</summary>
    </entry>

</feed>
```

	<p>Atom 1.0 Feed Tags</p> <p>An Atom 1.0 Feed Document will be constructed of the following two elements:</p> <ul style="list-style-type: none"> • <u><feed></u> Elements • <u><entry></u> Elements
18.	<p>Summarize the advantage of RSS documents?</p> <p style="text-align: center;">RSS - Advantages</p> <p>RSS is taking off so quickly because people are liking it. RSS is easy to use and it has advantages for a publisher as well as for a subscriber. Here we have listed out a few advantages of RSS for subscribers as well as for publishers.</p> <h3>Advantages for Subscribers</h3> <p>RSS subscribers are the people who subscribe to read a published Feed. Here are some of the advantages of RSS Feeds for subscribers:</p> <ul style="list-style-type: none"> • All news at one place: You can subscribe to multiple news groups and then you can customize your reader to have all the news on a single page. It will save you a lot of time. • News when you want it: Rather than waiting for an e-mail, you go to your RSS reader when you want to read a news. Furthermore, RSS Feeds display more quickly than information on web-sites, and you can read them offline if you prefer. • Get the news you want: RSS Feed comes in the form of headlines and a brief description so that you can easily scan the headlines and click only those stories that interest you. • Freedom from e-mail overload: You are not going to get any email for any news or blog update. You just go to your reader and you will find updated news or blog automatically whenever there is a change on the RSS server. • Easy republishing: You may be both a subscriber and a publisher. For example, you may have a web-site that collects news from various other sites and then republishes it. RSS allows you to easily capture that news and display it on your site. <h3>Advantages for Publishers</h3> <p>RSS publishers are the people who publish their content through RSS feed. We would suggest you to use RSS:</p> <ul style="list-style-type: none"> • if you want to get your message out and easily, • if you want people to see what you publish, and • if you want your news to bring people back to your site. <p>Here are some of the advantages of RSS if you publish on the Web:</p>

	<ul style="list-style-type: none"> • Easier publishing: RSS is really simple publishing. You don't have to maintain a database of subscribers to send your information to them, instead they will access your Feed using a reader and will get updated content automatically. • A simpler writing process: If you have a new content on your web site, you only need to write an RSS Feed in the form of titles and short descriptions, and link back to your site. • An improved relationship with your subscribers: Because people subscribe from their side, they don't feel as if you are pushing your content on them. • The assurance of reaching your subscribers: RSS is not subject to spam filters, your subscribers get the Feeds, which they subscribe to and nothing more. • Links back to your site: RSS Feeds always include links back to a website. It directs a lot of traffic towards your website. • Relevance and timeliness: Your subscribers always have the latest information from your site.
19.	<p>Rewrite the declaration for elements in XML.</p>
	<p>How would you prepare the steps to get the RSS file on web?</p> <h3>Uploading an RSS Feed</h3> <p>Here are the simple steps to put your RSS Feed on the web.</p> <ul style="list-style-type: none"> • First decide which version of RSS Feed you are going to use for your site. We would recommend you to use the latest version available. • Create your RSS Feed in a text file with extension either .xml or .rdf. Upload this file on your web server. • You should validate your RSS Feed before making it live. Check the next chapter on RSS Feed Validation. • Create a link on your Web Pages for the RSS Feed file. You will use a small yellow button for the link that says either RSS or XML.
20.	<p>Now, your RSS Feed is online and people can start using it. But there are ways to promote your RSS Feed so that more number of people can use your RSS Feed.</p> <h3>Promote Your RSS Feed</h3> <ul style="list-style-type: none"> • Submit your RSS Feed to the RSS Feed Directories. There are many directories available on the web, where you can register your Feed. Some of them are given here: <ul style="list-style-type: none"> ○ <u>Syndic8</u>: Over 300,000 Feeds listed. ○ <u>Daypop</u>: Over 50,000 feeds listed. ○ <u>Newsisfree</u>: Over 18,000 Feeds. • Register your Feed with the major search engines. Similar to your web pages, you can add your Feed as well with the following major search engines. <ul style="list-style-type: none"> ○ Yahoo - http://publisher.yahoo.com/promote.php

- Google - <http://www.google.com/webmasters/add.html>
- Bing - <http://www.bing.com/toolbox/submit-site-url>

Keeping Up-To-Date Feed

As we have explained earlier, RSS Feed makes sense for the site which are changing their content very frequently, for example, any news or blogging sites.

So now, you have got RSS Feed buttons from Google, Yahoo, and MSN. You must make sure to update your content frequently and that your RSS Feed is constantly available.

PART-B

- (i) **Describe** about the introduction and installation of PHP.

Introduction to PHP

PHP is one of the most widely used server side scripting language for web development. Popular websites like Facebook, Yahoo, Wikipedia etc are developed using PHP.

PHP is so popular because it's very simple to learn, code and deploy on server, hence it has been the first choice for beginners since decades.

Uses of PHP

To further fortify your trust in PHP, here are a few applications of this amazing scripting language:

1. It can be used to **create Web applications** like Social Networks(Facebook, Digg), Blogs(Wordpress, Joomla), eCommerce websites(OpenCart, Magento etc.) etc.
2. **Common Line Scripting.** You can write PHP scripts to perform different operations on any machine, all you need is a PHP parser for this.
3. **Create Facebook applications** and easily integrate Facebook plugins in your website, using Facebook's PHP SDK. Check this [link](#) for more information.
4. **Sending Emails** or building email applications because PHP provides with a robust email sending function.
5. Wordpress is one of the most used blogging(CMS) platform in the World, and if you know PHP, you can try a hand in **Wordpress plugin development**.

Manual Installation

Step 1: Download the files. Download the latest **PHP 5 ZIP** package from www.php.net/downloads.php. ...

Step 2: Extract the files. ...

Step 3: Configure **php**. ...

Step 4: Add C:\php to the path environment variable. ...

Step 5: Configure **PHP** as an Apache module. ...

Step 6: Test a **PHP** file.

(ii) **Design** simple calculator using PHP.

Calculator.php

<!DOCTYPE html>

```
<html>
    <head>
        <title>Simple Calculator In PHP | Webdevtrick.com</title>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <link
            href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css"
            rel="stylesheet">
    </head>
    <body>

        <div class="container" style="margin-top: 50px">

            <?php

                // If the submit button has been pressed
                if(isset($_POST['submit']))
                {
                    // Check number values
                    if(is_numeric($_POST['number1']) && is_numeric($_POST['number2']))
                    {
                        // Calculate total
                        if($_POST['operation'] == 'plus')
                        {
                            $total = $_POST['number1'] + $_POST['number2'];
                        }
                        if($_POST['operation'] == 'minus')
                        {
                            $total = $_POST['number1'] - $_POST['number2'];
                        }
                        if($_POST['operation'] == 'multiply')
                        {
                            $total = $_POST['number1'] * $_POST['number2'];
                        }
                    }
                }
            </?php>

        </div>
    </body>
</html>
```

```

        }
        if($_POST['operation'] == 'divided by')
        {
            $total = $_POST['number1'] / $_POST['number2'];
        }

        // Print total to the browser
        echo "<h1>{$_POST['number1']} {$_POST['operation']}
```

{\$_POST['number2']} equals {\$total}</h1>";

```

    } else {

        // Print error message to the browser
        echo 'Numeric values are required';

    }
}
// end PHP. Code by webdevtrick.com
?>

<!-- Calculator form by webdevtrick.com -->
<form method="post" action="calculator.php">
    <input name="number1" type="text" class="form-control" style="width: 150px; display: inline" />
    <select name="operation">
        <option value="plus">Plus</option>
        <option value="minus">Minus</option>
        <option value="multiply">Multiply</option>
        <option value="divided by">Divide</option>
    </select>
    <input name="number2" type="text" class="form-control" style="width: 150px; display: inline" />
    <input name="submit" type="submit" value="Calculate" class="btn btn-primary" />
</form>

</div>

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

Explain about control statements and data types in PHP with example.

Control Statements in PHP with Examples

2.

Like any other languages, PHP is built out of a series of control statements. The control statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing or an empty statement.

In PHP we have the following conditional statements:

if statement – We use this control statement to execute some code only if a specified condition is true.

if...else statement – We use this control statement to execute some code if a condition is true and another code if the condition is false.

if...elseif....else statement – We use this control statement to select one of several blocks of code to be executed

switch statement – We use this control statement to select one of many blocks of code to be executed

1. The if Statement

Use the if statement to execute some code only if a specified condition is true. The expression is evaluated to its Boolean value. If expression evaluates to TRUE, PHP will execute statement, and if it evaluates to FALSE – it'll ignore it

Syntax

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

The following example would display " A is bigger than B" if \$a is bigger than \$b:

```
<?php  
if ($a > $b)  
echo "A is bigger than B";  
?>
```

2. The if...else Statement

elseif, as its name suggests, is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if expression evaluates to FALSE. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to TRUE.

```
if (condition)  
    code to be executed if condition is true;  
else  
    code to be executed if condition is false;
```

For example, the following code would display a is bigger than b, a equal to b or a is smaller than b:

```
<?php  
if ($a > $b) {  
    echo "a is bigger than b";  
} elseif ($a == $b) {  
    echo "a is equal to b";  
} else {  
    echo "a is smaller than b";  
}  
?>
```

3. The if...elseif....else Statement

4. The Switch Statement

The switch statement is similar to IF statements on the same expression. In many occasions,

Use the if...elseif...else statement to select one of several blocks of code to be executed.

```
if (condition)
    code to be executed if condition is true;
elseif (condition)
    code to be executed if condition is true;
else
    code to be executed if condition is false;
```

Note: Note that elseif and else if will only be considered exactly the same when using curly brackets as in the above example. When using a colon to define your if/elseif conditions, you must not separate else if into two words, or PHP will fail with a parse error.

you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the switch statement is for.

```
switch ()
{
    case condition1
        break;
    case condition2
        break;
}
```

For example, the following code would display \$i matched value as 0 or 1 or 2:

```
<?php
switch ($i) {
    case 0:
        echo "i equals 0";
    case 1:
        echo "i equals 1";
    case 2:
        echo "i equals 2";
}
?>
```

(i) **Create** an XML document that marks up various sports and their descriptions. Use XSLT to tabulate neatly the elements and attributes of the document.

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

```
-<events league="WC Falun" tournament="2010/2011" template="World Cup"
sport="Cross Country Skiing" ut="2012-09-05" id="821135">
```

3.

```
-<event id="866683" status="Finished" round="8001 - 1/1 (Final)" date="2011-03-20
13:15:00" name="10 km Freestyle Handicap Pursuit">
```

```
-<results participantname="Marit Bjoergen" participantid="43427">
```

```
<result id="9498426" value="1" type="rank"/>
```

```
<result id="9498424" value="27:58.0" type="duration"/>
```

```

<result id="9505038" value="200" type="points"/>

<result id="9498425" value="" type="comment"/>

<result id="9497448" value="1" type="startnumber"/>

</results>

-<results participantname="Justyna Kowalczyk" participantid="43775">

<result id="9498429" value="2" type="rank"/>

<result id="9498427" value="+1:58.0" type="duration"/>

<result id="9505039" value="160" type="points"/>

<result id="9498428" value="" type="comment"/>

<result id="9497454" value="2" type="startnumber"/>

</results>
</event></events>
```

(ii) **Illustrate** a JSP page that enables the user to input the first name and in response outputs the last name.

POST Method Example Using Form

Below is the **main.jsp** JSP program to handle the input given by web browser using the GET or the POST methods.

```

<html>
  <head>
    <title>Using GET and POST Method to Read Form Data</title>
  </head>

  <body>
    <center>
      <h1>Using POST Method to Read Form Data</h1>

      <ul>
        <li><p><b>First Name:</b>
          <%= request.getParameter("first_name")%>
        </p></li>
        <li><p><b>Last Name:</b>
          <%= request.getParameter("last_name")%>
        </p></li>
      </ul>
```

```
</body>  
</html>
```

Following is the content of the **Hello.htm** file –

```
<html>  
  <body>  
  
    <form action = "main.jsp" method = "POST">  
      First Name: <input type = "text" name = "first_name">  
      <br />  
      Last Name: <input type = "text" name = "last_name" />  
      <input type = "submit" value = "Submit" />  
    </form>  
  
  </body>  
</html>
```

Let us now keep **main.jsp** and **hello.htm** in **<Tomcat-installationdirectory>/webapps/ROOT directory**. When you access **http://localhost:8080/Hello.htm**, you will receive the following output.

First Name:
Last Name:

Try to enter the First and the Last Name and then click the submit button to see the result on your local machine where tomcat is running.

4. **Create** a webserver based chat application using PHP. The application should provide the following functions Login, Send message (to one or more contacts) and Receive messages (from one or more contacts)

5. (i) **Write** a PHP program that tests whether an email address is input correctly. Test your program with both valid and invalid email addresses.

PHP - Validate Name, E-mail, and URL

Example

```
<?php  
// define variables and set to empty values
```

```

$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
        // check if name only contains letters and whitespace
        if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
            $nameErr = "Only letters and white space allowed";
        }
    }

    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = test_input($_POST["email"]);
        // check if e-mail address is well-formed
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = "Invalid email format";
        }
    }

    if (empty($_POST["website"])) {
        $website = "";
    } else {
        $website = test_input($_POST["website"]);
        // check if URL address syntax is valid (this regular expression
        // also allows dashes in the URL)
        if (!preg_match("/\b(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\%/?=~_|!:,.;]*[-a-z0-9+&@#\%=~_|]/i",$website)) {
            $websiteErr = "Invalid URL";
        }
    }

    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = test_input($_POST["comment"]);
    }

    if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
    } else {
        $gender = test_input($_POST["gender"]);
    }
}

```

```
}
```

```
?>
```

OUTPUT

PHP Form Validation Example

* required field

Name: *

E-mail: *

Website:

Comment:

Gender: Female Male Other *

Your Input:

Anand M
anand@ibm.com
www.ibm.com
Yhis is a comment
male

Identify and explain about database connectivity illustrate PHP connectivity with any of the databases.

6.

METHOD FOR: CONNECTING TO MYSQL USING MYSQL

The MySQL Improved extension uses the *mysqli* class, which replaces the set of legacy MySQL functions.

To connect to MySQL using the MySQL Improved extension, follow these steps:

1. Use the following PHP code to connect to MySQL and select a database. Replace *username* with your username, *password* with your password, and *dbname* with the database name:

```
2.    <?php  
3.        $mysqli = new mysqli("localhost", "username", "password", "dbname");  
    ?>
```

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = mysqli_connect($servername, $username, $password);  
  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli_connect_error());  
}  
echo "Connected successfully";  
?>
```

- (i) Discuss on methods for using cookies in PHP.

Cookies in PHP

Cookies are used to store the information of a web page in a remote browser, so that when the same user comes back to that page, that information can be retrieved from the browser itself.

7. **Uses of cookie**

Cookies are often used to perform following tasks:

- **Session management:** Cookies are widely used to manage user sessions. For example, when you use an online shopping cart, you keep adding items in the cart and finally when you checkout, all of those items are added to the list of items you have purchased. This can be achieved using cookies.
- **User identification:** Once a user visits a webpage, using cookies, that user can be remembered. And later on, depending upon the search/visit pattern of the user,

content which the user likely to be visited are served. A good example of this is 'Retargetting'. A concept used in online marketing, where depending upon the user's choice of content, advertisements of the relevant product, which the user may buy, are served.

- **Tracking / Analytics:** Cookies are used to track the user. Which, in turn, is used to analyze and serve various kind of data of great value, like location, technologies (e.g. browser, OS) form where the user visited, how long (s)he stayed on various pages etc.

How to create a cookie in PHP

PHP has a setcookie() function to send a cookie. We will discuss this function in detail now.

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

setcookie() returns boolean.

Example:

Following example shows how to create a cookie in PHP.

```
<?php  
  
$cookie_value = "w3resource tutorials";  
  
setcookie("w3resource", $cookie_value, time()+3600, "/home/your_username/",  
"example.com", 1, 1);  
  
if (isset($_COOKIE['cookie']))  
  
echo $_COOKIE["w3resource"];  
  
?>
```

(ii) **Give a note on regular expressions.**

Summarize in detail the XML schema, built in and user defined data types.

What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

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

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

XSD Simple Elements

XML Schemas define the elements of your XML files.

A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.

The text can be of many different types like boolean, string, date, etc.), or it can be a custom type that you can define yourself.

You can also add restrictions (facets) to a data type in order to limit its content, or you can require the data to match a specific pattern.

The syntax for defining a simple element is:

```
<xs:element name="xxx" type="yyy"/>
```

where xxx is the name of the element and yyy is the data type of the element.

XML Schema has a lot of built-in data types. The most common types are:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Example

Simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Here are some XML elements:

```
<lastname>Ronald</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

Simple elements may have a default value which is automatically assigned to the element when no other value is specified as shown below:

```
<xs:element name="color" type="xs:string" default="red"/>
```

How to Define a Complex Element

We can define a complex element in an XML Schema two different ways:

1. The "employee" element can be declared directly by naming the element, like this:

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2. The "employee" element can have a type attribute that refers to the name of the complex type to use:

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

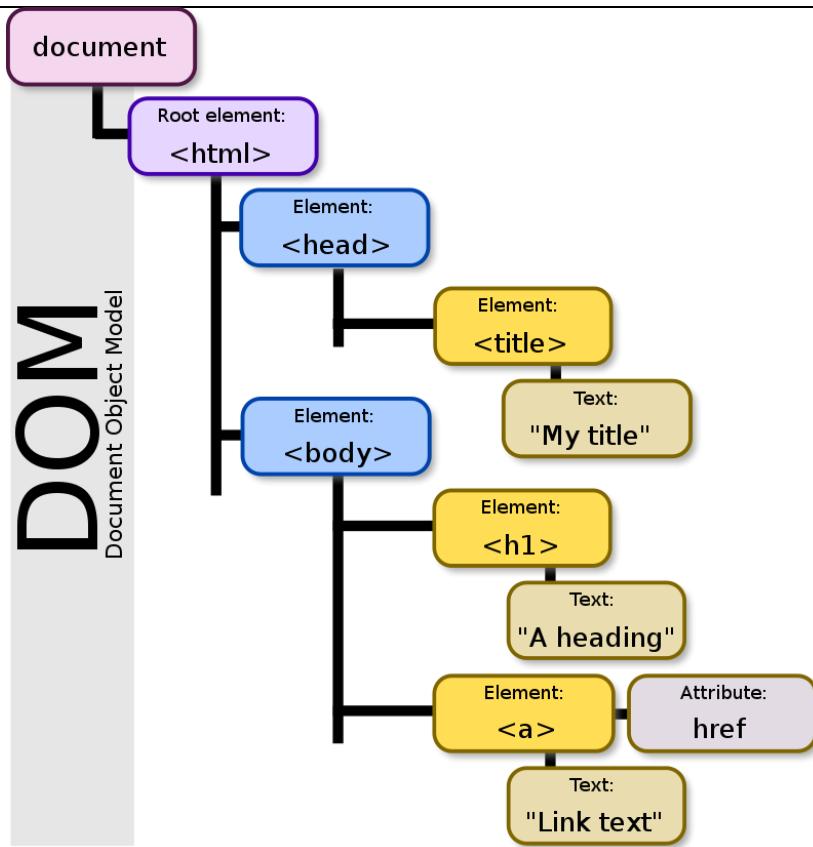
If you use the method described above, several elements can refer to the same complex type, like this:

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

(i) **Demonstrate** the building blocks of DOM.

9. The **Document Object Model (DOM)** is a [cross-platform](#) and [language](#)-independent interface that treats an [XML](#) or [HTML](#) document as a [tree structure](#) wherein each [node](#) is an [object](#) representing a part of the document. The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document. Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.



(ii) Classify the types of DTD.

XML DTD

An XML document with correct syntax is called "Well Formed".

An XML document validated against a DTD is both "Well Formed" and "Valid".

DTD stands for Document Type Definition.

A DTD defines the structure and the legal elements and attributes of an XML document.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

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

The DOCTYPE declaration above contains a reference to a DTD file. The content of the DTD file is shown and explained below.

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

Note.dtd:

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

The DTD above is interpreted like this:

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

Tip: #PCDATA means parseable character data.

How do you **infer the significant** differences between DID and XML schema for defining XML document structures with appropriate examples.

Difference Between XML Schema and DTD

10.

XML Schema vs. DTD

DTD, or Document Type Definition, and XML Schema, which is also known as XSD, are two ways of describing the structure and content of an XML document. DTD is the older

of the two, and as such, it has limitations that XML Schema has tried to improve.

Summary:

1. XML Schema is namespace aware, while DTD is not.
2. XML Schemas are written in XML, while DTDs are not.
3. XML Schema is strongly typed, while DTD is not.
4. XML Schema has a wealth of derived and built-in data types that are not available in DTD.
5. XML Schema does not allow inline definitions, while DTD does.

(i) List out data types data types of XML

XML Schema Data Types

XML Schema data types can be generally categorized a "simple type" (including embedded simple type) and "complex type." The "embedded simple type" is already defined, but can be used to create a new type through restriction or extension.

Table : XML Schema Data Types

11.	Simple Type	User can independently define. This type is used when a restriction is placed on an embedded simple type to create and use a new type.
	Complex Type	User can independently define. This type is used when the type has a child element or attribute.

A simple type is a type that only contains text data. This type can be used with element declarations and attribute declarations. On the other hand, a complex data type is a type that has a child element or attribute structure.

•Simple Type Example

```
<xs:element name="Department" type="xs:string" />
```

Here, the section described together with "xs:string" is an embedded simple type according to XML Schema. In this example, we have established the definition that the data type for the element called "Department" is a text string.

•Complex Type Example

```
<xs:complexType name="EmployeeType">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Name" />
    <xs:element ref="Department" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Name" type="xs:string" />
<xs:element name="Department" type="xs:string" />
```

(ii) **Explain** about the attributes of XML.

XML Elements vs. Attributes

Take a look at these examples:

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

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

In the first example gender is an attribute. In the last, gender is an element. Both examples provide the same information.

There are no rules about when to use attributes or when to use elements in XML.

Summarize on the following
(i) DOM based Parsing.

12.

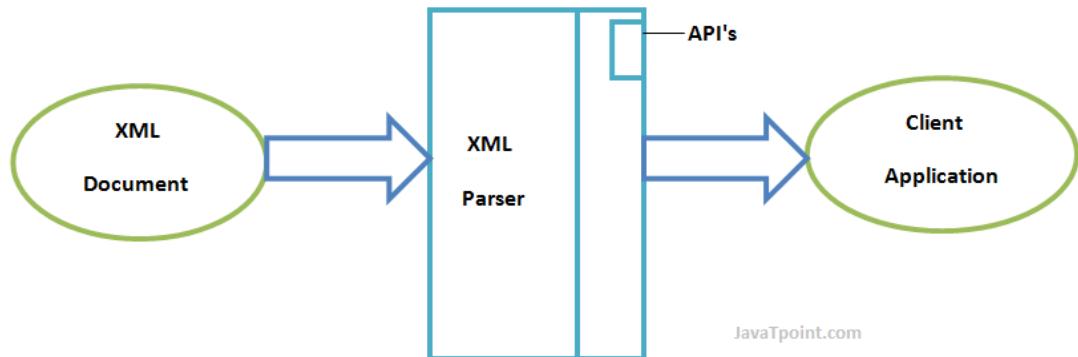
(ii) SAX based Parsing.

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

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

Let's understand the working of XML parser by the figure given below:



JavaTpoint.com

Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

DOM (Document Object Model)

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

Features of DOM Parser

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

DOM Parser has a tree based structure.

Advantages

- | | |
|--|--|
| | <ol style="list-style-type: none"> 1) It supports both read and write operations and the API is very simple to use. 2) It is preferred when random access to widely separated parts of a document is required. |
|--|--|

Disadvantages

- 1) It is memory inefficient. (consumes more memory because the whole XML document needs to be loaded into memory).
 - 2) It is comparatively slower than other parsers.
-

SAX (Simple API for XML)

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

Features of SAX Parser

It does not create any internal structure.

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

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

Advantages

- 1) It is simple and memory efficient.
- 2) It is very fast and works for huge documents.

Disadvantages

- 1) It is event-based so its API is less intuitive.
- 2) Clients never know the full information because the data is broken into pieces.

	(i) Compare and contrast RSS & ATOM.
--	---

 13. |

Difference Between RSS and ATOM

RSS vs ATOM

- Really Simple Syndication or RSS has been the standard for web feeds for a considerable time.
- Web feeds contains either a summary or the full text content of a web page.
- The problem with RSS is the often confusing and non standard conventions used by RSS due in part to its scattered development.
- The advent of the ATOM syndication standard was a response to the design flaws of the RSS standard.
- The primary advantage of the ATOM is its adaptation as the IETF standard.
- Being an IETF standard, each atom feed contains an explicit declaration of the format of the content along with what language is used.
- RSS feeds do not declare its content, but since it only contains plain text or escaped [HTML](#), it is rather easy for the browser to distinguish which is which.

A major flaw of RSS is in its code. RSS code isn't really very usable in other [XML](#) vocabularies since it wasn't really intended to do so at the very beginning. ATOM code has been built from the ground with modularity in mind. Therefore, a great majority of its code is reusable even with other XML vocabularies like RSS.

Summary:

1. ATOM is an IETF standard while RSS is not
2. ATOM feeds explicitly indicates the content while the browser is left to figure out whether the RSS feed contains plain text or escaped HTML
3. ATOM code is modular and reusable while RSS code is not
4. RSS still holds dominance in the syndication format due to its head start and popularity

(ii) **Explain** in detail about XSL elements.

XSLT <xsl:element>

Definition and Usage

The <xsl:element> element is used to create an element node in the output document.

Syntax

```
<xsl:element
  name="name"
  namespace="URI"
  use-attribute-sets="namelist">

  <!-- Content:template -->

</xsl:element>
```

Attributes

Attribute	Value	Description
name	name	Required. Specifies the name of the element to be created (the value of the name attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{\$country}" />)
namespace	URI	Optional. Specifies the namespace URI of the element (the value of the namespace attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{\$country}" namespace="{\$someuri}" />)
use-attribute-sets	namelist	Optional. A white space separated list of attribute-sets containing attributes to be added to the element

Example 1

Create a "singer" element that contains the value of each artist element:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <xsl:for-each select="catalog/cd">
      <xsl:element name="singer">
        <xsl:value-of select="artist" />
      </xsl:element>
      <br />
    </xsl:for-each>
  </xsl:template>

</xsl:stylesheet>
```

EXAMPLE OUTPUT FILES

XML File

```
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
</catalog>
```

XSL File

```
xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <body>
        <h2>My CD Collection</h2>
        <xsl:for-each select="catalog/cd">
```

```

<xsl:element name="singer">
<xsl:value-of select="artist"/>
</xsl:element>
<br/>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

RESULT

My CD Collection

Bob Dylan
 Bonnie Tyler
 Dolly Parton
 Gary Moore

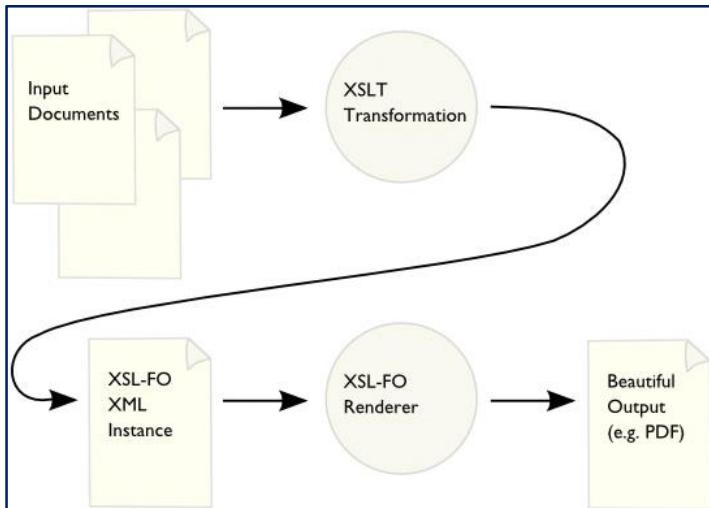
Explain in detail about
 (i) XSL and XSLT transformation

XSLT - Transformation

What is XSLT?

[XSL Transformations](#) (XSLT 2.0) is a language for transforming XML documents into other XML documents, text documents or HTML documents. You might want to format a chapter of a book using [XSL-FO](#), or you might want to take a database query and format it as HTML.

14.



Wildly Popular

XSLT has become the language of choice for a very wide range of XML applications. It is of course still used to produce XSL-FO documents for printing, but it is also used to integrate back-end software for Web sites. We can find XSLT inside most modern Web browsers, so

that XML can be transformed on the fly without the user even noticing; you will find XSLT on the desktop, in servers, in network appliances.

What is XSLT Used For?

If you make a purchase on eBay, or buy a book at Amazon, chances are that pretty much everything you see on every Web page has been processed with XSLT. Use XSLT to process multiple XML documents and to produce any combination of text, HTML and XML output. XSLT support is shipped with all major computer operating systems today, as well as being built in to all major Web browsers.

XSLT – Transformation : STEPS

- 1) Start with a Raw XML Document
- 2) Create an XSL Style Sheet
- 3) Link the XSL Style Sheet to the XML Document

1) Start with a Raw XML Document

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

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

2) Create an XSL Style Sheet

Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template:

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

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

  <xsl:template match="/">
    <html>
      <body>
        <h2>My CD Collection</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Title</th>
            <th>Artist</th>
          </tr>
```

```

<xsl:for-each select="catalog/cd">
<tr>
  <td><xsl:value-of select="title"/></td>
  <td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>

</xsl:stylesheet>

```

3) Link the XSL Style Sheet to the XML Document

Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>

    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
</catalog>

```

If you have an XSLT compliant browser it will nicely **transform** your XML into XHTML.

Sample OUTPUT

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore

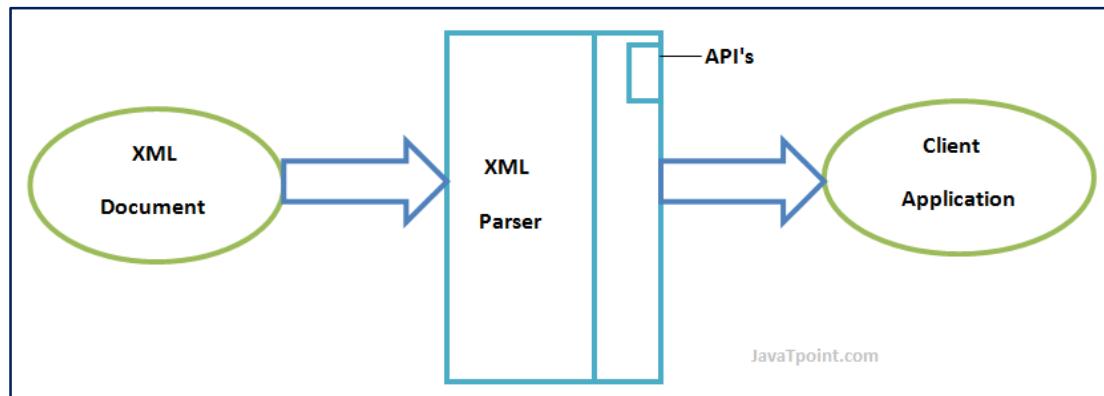
(ii) Comparison of DOM & SAX

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

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

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

These are the two main types of XML Parsers:

3. DOM
4. SAX

DOM (Document Object Model)

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

Features of DOM Parser

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

DOM Parser has a tree based structure.

Advantages

- 1) It supports both read and write operations and the API is very simple to use.
- 2) It is preferred when random access to widely separated parts of a document is required.

Disadvantages

- 1) It is memory inefficient. (consumes more memory because the whole XML document needs to be loaded into memory).
 - 2) It is comparatively slower than other parsers.
-

SAX (Simple API for XML)

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

Features of SAX Parser

It does not create any internal structure.

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

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

Advantages

- 1) It is simple and memory efficient.
- 2) It is very fast and works for huge documents.

Disadvantages

- 1) It is event-based so its API is less intuitive.
- 2) Clients never know the full information because the data is broken into pieces.

PART-C

1. **Explain how you shall carry out String Manipulations using a PHP Program.**

Manipulating PHP Strings

PHP provides many built-in functions for manipulating strings like calculating the length of a string, find substrings or characters, replacing part of a string with different characters, take a string apart, and many others.

Here are the examples of some of these functions.

Calculating the Length of a String

The `strlen()` function is used to calculate the number of characters inside a string. It also includes the blank spaces inside the string.

Example

[Run this code »](#)

```
<?php  
$my_str = 'Welcome to Tutorial Republic';  
  
// Outputs: 28  
echo strlen($my_str);  
?>
```

Counting Number of Words in a String

The `str_word_count()` function counts the number of words in a string.

Example

[Run this code »](#)

```
<?php  
$my_str = 'The quick brown fox jumps over the lazy dog.';  
  
// Outputs: 9  
echo str_word_count($my_str);  
?>
```

Replacing Text within Strings

The `str_replace()` replaces all occurrences of the search text within the target string.

Example

[Run this code »](#)

```
<?php  
$my_str = 'If the facts do not fit the theory, change the facts.';  
  
// Display replaced string  
echo str_replace("facts", "truth", $my_str);  
?>
```

The output of the above code will be:

If the truth do not fit the theory, change the truth.

You can optionally pass the fourth argument to the `str_replace()` function to know how many times the string replacements was performed, like this.

Example

[Run this code »](#)

```
<?php  
$my_str = 'If the facts do not fit the theory, change the facts.';  
  
// Perform string replacement  
str_replace("facts", "truth", $my_str, $count);  
  
// Display number of replacements performed  
echo "The text was replaced $count times.";  
?>
```

The output of the above code will be:

The text was replaced 2 times.

Reversing a String

The `strrev()` function reverses a string.

Example

[Run this code »](#)

```
<?php  
$my_str = 'You can do anything, but not everything.';  
  
// Display reversed string  
echo strrev($my_str);  
?>
```

	<p>The output of the above code will be:</p> <p>.gnihtyreve ton tub ,gnihtyna od nac uoY</p>								
2.	<p>Design a PHP application for College Management System with appropriate built-in functions and database.</p>								
	<p>Design application to send an email using PHP.</p> <h3><u>PHP mail() Function</u></h3> <h3>Example</h3> <p>Send a simple email:</p> <pre><?php // the message \$msg = "First line of text\nSecond line of text"; // use wordwrap() if lines are longer than 70 characters \$msg = wordwrap(\$msg,70); // send email mail("someone@example.com","My subject",\$msg); ?></pre>								
3.	<h3>Syntax</h3> <p><code>mail(<i>to,subject,message,headers,parameters</i>);</code></p> <h3>Parameter Values</h3> <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>to</i></td> <td>Required. Specifies the receiver / receivers of the email</td> </tr> <tr> <td><i>subject</i></td> <td>Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters</td> </tr> <tr> <td><i>message</i></td> <td>Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. Windows note: If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php</td> </tr> </tbody> </table>	Parameter	Description	<i>to</i>	Required. Specifies the receiver / receivers of the email	<i>subject</i>	Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters	<i>message</i>	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. Windows note: If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php
Parameter	Description								
<i>to</i>	Required. Specifies the receiver / receivers of the email								
<i>subject</i>	Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters								
<i>message</i>	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. Windows note: If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php								

		\$txt = str_replace("\n.", "\n..", \$txt); ?>	
<i>headers</i>		Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n). Note: When sending an email, it must contain a From header. This can be set with this parameter or in the php.ini file.	
<i>parameters</i>		Optional. Specifies an additional parameter to the sendmail program (the one defined in the sendmail_path configuration setting). (i.e. this can be used to set the envelope sender address when using sendmail with the -f sendmail option)	

Technical Details

Return Value:	Returns the hash value of the <i>address</i> parameter, or FALSE on failure. Note: Keep in mind that even if the email was accepted for delivery, it does NOT mean the email is actually sent and received!
PHP Version:	4+
PHP Changelog:	PHP 7.2: The headers parameter also accepts an array PHP 5.4: Added header injection protection for the <i>headers</i> parameter. PHP 4.3.0: (Windows only) All custom headers (like From, Cc, Bcc and Date) are supported, and are not case-sensitive. PHP 4.2.3: The <i>parameter</i> parameter is disabled in safe mode PHP 4.0.5: The <i>parameter</i> parameter was added

More Examples

Send an email with extra headers:

```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From: webmaster@example.com" . "\r\n" .
"CC: somebodyelse@example.com";

mail($to,$subject,$txt,$headers);
?>
```

Send an HTML email:

```
<?php
$to = "somebody@example.com, somebodyelse@example.com";
$subject = "HTML email";

$message = "
<html>
<head>
<title>HTML email</title>
</head>
<body>
<p>This email contains HTML Tags!</p>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
</tr>
</table>
</body>
</html>
";

// Always set content-type when sending HTML email
$headers = "MIME-Version: 1.0" . "\r\n";
$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

// More headers
$headers .= 'From: <webmaster@example.com>' . "\r\n";
$headers .= 'Cc: myboss@example.com' . "\r\n";

mail($to,$subject,$message,$headers);
?>
```

Summarize about XML schema and XML Parsers and Validation.

XML Schema

4. What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

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

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

Why Learn XML Schema?

In the XML world, hundreds of standardized XML formats are in daily use.

Many of these XML standards are defined by XML Schemas.

XML Schema is an XML-based (and more powerful) alternative to DTD.

XML Parser

All major browsers have a built-in XML parser to access and manipulate XML.

The [XML DOM \(Document Object Model\)](#) defines the properties and methods for accessing and editing XML.

However, before an XML document can be accessed, it must be loaded into an XML DOM object.

All modern browsers have a built-in XML parser that can convert text into an XML DOM object.

Parsing a Text String

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

Example

```
<html>
<body>

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

<script>
var text, parser, xmlDoc;

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

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

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

</body>
</html>
```

OUTPUT

Everyday Italian

XML - Validation

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration(DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are –

- Well-formed XML document
- Valid XML document

Well-formed XML Document

An XML document is said to be **well-formed** if it adheres to the following rules –

- Non DTD XML files must use the predefined character entities for **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)**.
- It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
- Each of its opening tags must have a closing tag or it must be a self ending tag.(<title>....</title> or <title/>).
- It must have only one attribute in a start tag, which needs to be quoted.
- **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)** entities other than these must be declared.

Example

Following is an example of a well-formed XML document –

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address
[
    <!ELEMENT address (name,company,phone)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT company (#PCDATA)>
    <!ELEMENT phone (#PCDATA)>
]>

<address>
    <name>Tanmay Patil</name>
    <company>TutorialsPoint</company>
    <phone>(011) 123-4567</phone>
</address>
```

The above example is said to be well-formed as –

- It defines the type of document. Here, the document type is **element** type.
- It includes a root element named as **address**.
- Each of the child elements among name, company and phone is enclosed in its self explanatory tag.
- Order of the tags is maintained.

Valid XML Document

If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.

CS8651 -UNIT 5-NOTES

CS8651- Internet Programming 2017Reg UNIT V - INTRODUCTION TO AJAX and WEB SERVICES	
PART-A	
No	Questions
1.	<p>Describe AJAX Control Extender Toolkit.</p> <p>What is the ASP.NET AJAX Control Toolkit?</p> <p>The ASP.NET AJAX Control Toolkit is an open-source project built on top of the Microsoft ASP.NET AJAX framework. It is a joint effort between Microsoft and the ASP.NET AJAX community that provides a powerful infrastructure to write reusable, customizable and extensible ASP.NET AJAX extenders and controls, as well as a rich array of controls that can be used out of the box to create an interactive Web experience.</p> <p>They are designed using concepts that are familiar to ASP.NET Web Forms application developers. Using the Ajax Control Toolkit, you can build Ajax-enabled ASP.NET Web Forms applications and ASP.NET MVC Web applications by dragging the controls from the Visual Studio Toolbox onto a page. The Ajax Control Toolkit is an open-source project that is part of the CodePlex Foundation</p> <p>The AJAX Control Toolkit contains more than 30 controls that enable you to easily create rich, interactive web pages.</p>
2.	<p>Discuss the advantages of AJAX.</p> <p><u>Advantages of AJAX</u></p> <ul style="list-style-type: none"> • Reduce the traffic travels between the client and the server. • Response time is faster so increases performance and speed. • You can use JSON (JavaScript Object Notation) which is alternative to XML. JSON is key value pair and works like an array. • You can use Firefox browser with an add-on called as Firebug to debug all Ajax calls. • Ready Open source JavaScript libraries available for use – JQuery, Prototype, Scriptaculous, etc.. • AJAX communicates over HTTP Protocol.
3.	<p>Identify the role of a callback function in performing a partial page update in an AJAX application.</p> <p><u>Partial-page rendering with UpdatePanels</u></p>

One of the most fascinating controls in the ASP.NET AJAX framework is the *UpdatePanel*. This new control replaces the need for a page to refresh during a postback. Only portions of a page, designated by the UpdatePanel, are updated. This technique is known as *partial-page rendering* and can be highly effective in improving the user experience.

6.1.1. Evolution of the UpdatePanel

For years, programming with the XMLHttpRequest object has been the most commonly used approach for communicating with the server from client-side script. The complexities involved in coding those types of applications scared away a lot of developers. To assist, the overall scripting model in ASP.NET 2.0 was significantly enhanced to introduce the idea of *script callbacks*—a way for server controls to communicate with client-side scripts between callbacks. This model was powerful because it offered access to the state of all the controls on the page during a callback. Unfortunately, many developers found the model difficult to work with, and numerous concerns were raised. The lack of support for passing complex types as parameters to the server (only strings were allowed) made the prototype too rigid and exposed its limitations. Developers began to look elsewhere for solutions.

In an effort to address these concerns, members of the ASP.NET team began work on a communication library built on top of the callbacks. The primary objective of the library was to simplify the use of callbacks and to provide a rich set of APIs for enabling the exchange of complex and simple types between the server and client. From this library came a control called the *RefreshPanel*. The purpose of the RefreshPanel was to offer a server control that refreshed the contents of a page without a page refresh. Out of this hard work, the UpdatePanel emerged, with deeper integration into the page lifecycle and a more transparent footprint on the page.

NOTE

A *callback* is a piece of code that is passed in as a parameter or argument to other code. The other piece of code can call the callback code (usually a function) at any time, even numerous times, in response to some processing.

4.	<p>Differentiate AJAX forms with HTML5 forms.</p> <p>AJAX is the name of a communication architecture between web pages and server side.</p> <p>JQuery is a javascript library that is written for unifying JS method calls (regarding DOM manipulation, String and Array functions, DOM queries...etc.) in all browsers.</p> <p>HTML5 is the rendering specification to be implemented by all browser providers. For this rendering to work JS Engine should also be updated, so HTML5 also means a JS engine with new features like drawing on a canvas.</p>										
5.	<p>What is XML Http Request object? List its properties.</p> <h3><u>The XMLHttpRequest Object</u></h3> <p>With the XMLHttpRequest object you can update parts of a web page, without reloading the whole page.</p> <h3>The XMLHttpRequest Object</h3> <p>The XMLHttpRequest object is used to exchange data with a server behind the scenes.</p> <p>The XMLHttpRequest object is the developers dream, because you can:</p> <ul style="list-style-type: none"> • Update a web page without reloading the page • Request data from a server after the page has loaded • Receive data from a server after the page has loaded • Send data to a server in the background <p>XMLHttpRequest Object Methods</p> <table border="1"> <thead> <tr> <th>Method</th><th>Description</th></tr> </thead> <tbody> <tr> <td>abort()</td><td>Cancels the current request</td></tr> <tr> <td>getAllResponseHeaders()</td><td>Returns header information</td></tr> <tr> <td>getResponseHeader()</td><td>Returns specific header information</td></tr> <tr> <td>open(method,url,async,uname,pswd)</td><td>Specifies the type of request, the URL, if the request should be handled asynchronously or not, and other optional attributes of a request</td></tr> </tbody> </table>	Method	Description	abort()	Cancels the current request	getAllResponseHeaders()	Returns header information	getResponseHeader()	Returns specific header information	open(method,url,async,uname,pswd)	Specifies the type of request, the URL, if the request should be handled asynchronously or not, and other optional attributes of a request
Method	Description										
abort()	Cancels the current request										
getAllResponseHeaders()	Returns header information										
getResponseHeader()	Returns specific header information										
open(method,url,async,uname,pswd)	Specifies the type of request, the URL, if the request should be handled asynchronously or not, and other optional attributes of a request										

		method: the type of request: GET or POST url: the location of the file on the server async: true (asynchronous) or false (synchronous)	
	send(string)	send(string) Sends the request off to the server. string: Only used for POST requests	
	setRequestHeader()	Adds a label/value pair to the header to be sent	

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number (e.g. "404" for "Not Found" or "200" for "OK")
statusText	Returns the status-text (e.g. "Not Found" or "OK")

6.	<p>Summarize the need of SOAP and show its structure.</p> <p><u>XML Soap</u></p> <ul style="list-style-type: none"> • SOAP stands for Simple Object Access Protocol • SOAP is an application communication protocol • SOAP is a format for sending and receiving messages • SOAP is platform independent
----	---

	<ul style="list-style-type: none"> • SOAP is based on XML • SOAP is a W3C recommendation <p>Why SOAP?</p> <p>It is important for web applications to be able to communicate over the Internet.</p> <p>The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.</p> <p>A SOAP message is an ordinary XML document containing the following elements:</p> <ul style="list-style-type: none"> • An Envelope element that identifies the XML document as a SOAP message • A Header element that contains header information • A Body element that contains call and response information • A Fault element containing errors and status information
7.	<p>Can you develop the service end point interface in RPC?</p> <p><i>Example</i></p> <p>Sample Java Beans service endpoint implementation and interface</p> <p>The following example illustrates a simple explicit Java Beans service endpoint implementation and the associated service endpoint interface.</p> <pre>/** This is an excerpt from the service implementation file, EchoServicePortTypeImpl.java package com.ibm.wssample.echo; import java.io.ByteArrayInputStream; import java.io.ByteArrayOutputStream; import javax.xml.bind.JAXBContext; import javax.xml.bind.Marshaller; import javax.xml.bind.Unmarshaller; import javax.xml.transform.stream.StreamSource;</pre> <pre>@javax.jws.WebService(serviceName = "EchoService", endpointInterface = "com.ibm.wssample.echo.EchoServicePortType", targetNamespace="http://com/ibm/was/wssample/echo/", portName="EchoServicePort") public class EchoServicePortTypeImpl implements EchoServicePortType { public EchoServicePortTypeImpl() { } public String invoke(String obj) { String str; str = obj; return str;</pre>

```

        }

}

/** This is a sample EchoServicePortType.java service interface */
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebResult;
import javax.jws.WebService;
import javax.xml.ws.*;

@WebService(name = "EchoServicePortType",
            targetNamespace =
"http://com.ibm/was/wssample/echo/",
            wsdlLocation="WEB-INF/wsdl/Echo.wsdl")
public interface EchoServicePortType {

    /** ...the method process ...*/
    @WebMethod
    @WebResult(name = "response", targetNamespace =
"http://com.ibm/was/wssample/echo/")
    @RequestWrapper(localName = "invoke", targetNamespace =
"http://com.ibm/was/wssample/echo/", className =
"com.ibm.was.wssample.echo.Invoke")
    @ResponseWrapper(localName = "echoStringResponse",
targetNamespace = "http://com.ibm/was/wssample/echo/", className =
"com.ibm.was.wssample.echo.EchoStringResponse")
    public String invoke(
        @WebParam(name = "arg0", targetNamespace =
"http://com.ibm/was/wssample/echo/")
        String arg0);

}

```

Sample Provider endpoint implementation

The following example illustrates a simple Provider service endpoint interface for a Java class.

```

package jaxws.provider.source;
import javax.xml.ws.Provider;
import javax.xml.ws.WebServiceProvider;
import javax.xml.transform.Source;

@WebServiceProvider() public class SourceProvider implements
Provider<Source> {

    public Source invoke(Source data) {
        return data;
    }
}

```

8. **List any four examples of web services.**
A Web Service Example

In the following example we will use ASP.NET to create a simple Web Service that converts the temperature from Fahrenheit to Celsius, and vice versa:

```
<%@ WebService Language="VBScript" Class="TempConvert" %>

Imports System
Imports System.Web.Services

Public Class TempConvert :Inherits WebService

<WebMethod()> Public Function FahrenheitToCelsius(ByVal
Fahrenheit As String) As String
    dim fahr
    fahr=trim(replace(Fahrenheit,",","."))
    if fahr="" or IsNumeric(fahr)=false then return "Error"
    return (((fahr) - 32) / 9) * 5
end function

<WebMethod()> Public Function CelsiusToFahrenheit(ByVal Celsius
As String) As String
    dim cel
    cel=trim(replace(Celsius,",","."))
    if cel="" or IsNumeric(cel)=false then return "Error"
    return (((cel) * 9) / 5) + 32
end function

end class
```

This document is saved as an .asmx file. This is the ASP.NET file extension for XML Web Services.

Put the Web Service on Your Web Site

Using a form and the HTTP POST method, you can put the web service on your site, like this:

Fahrenheit to Celsius:	<input type="text"/>	<input type="button" value="Submit"/>
------------------------	----------------------	---------------------------------------

Celsius to Fahrenheit:	<input type="text"/>	<input type="button" value="Submit"/>
------------------------	----------------------	---------------------------------------

code to add the Web Service to a web page:

```

<form action='tempconvert.asmx/FahrenheitToCelsius'
method="post" target="_blank">
<table>
  <tr>
    <td>Fahrenheit to Celsius:</td>
    <td>
      <input class="frmInput" type="text" size="30" name="Fahrenheit">
    </td>
  </tr>
  <tr>
    <td></td>
    <td align="right">
      <input type="submit" value="Submit" class="button">
    </td>
  </tr>
</table>
</form>

<form action='tempconvert.asmx/CelsiusToFahrenheit'
method="post" target="_blank">
<table>
  <tr>
    <td>Celsius to Fahrenheit:</td>
    <td>
      <input class="frmInput" type="text" size="30" name="Celsius">
    </td>
  </tr>
  <tr>
    <td></td>
    <td align="right">
      <input type="submit" value="Submit" class="button">
    </td>
  </tr>
</table>
</form>

```

Substitute the "tempconvert.asmx" with the address of your web service like:

<http://www.example.com/xml/tempconvert.asmx>

9. **Discover** an example for web service registry along with its functions.
- Web Services Discovery** provides access to software systems over the Internet using standard protocols. In the most basic scenario there is a *Web Service Provider* that publishes a service and a *Web Service Consumer* that uses this service. Web Service Discovery is the process of finding suitable [web services](#) for a given task.^[1]
- Publishing a web service involves creating a [software artifact](#) and making it accessible to potential consumers. Web service providers augment a [service endpoint](#)

	<p><u>interface</u> with an interface description using the <u>Web Services Description Language</u> (WSDL) so that a consumer can use the service.</p> <p>Universal Description, Discovery, and Integration (UDDI) is an XML-based registry for business internet services. A provider can explicitly register a service with a <i>Web Services Registry</i> such as UDDI or publish additional documents intended to facilitate discovery such as <u>Web Services Inspection Language</u> (WSIL) documents. The service users or consumers can search web services manually or automatically. The implementation of UDDI servers and WSIL engines should provide simple search APIs or web-based <u>GUI</u> to help find Web services.</p> <p>Web services may also be discovered using <u>multicast</u> mechanisms like <u>WS-Discovery</u>, thus reducing the need for centralized registries in smaller networks.</p>
10.	<p>Analyze the need for web service.</p> <p>We should use web services as it comes with various advantages listed below</p> <p>Re-usability</p> <p>Once we develop some business logic, we can make it reuse for other applications</p> <p>Example: If 10 different applications requires to use our logic We can expose our logic over a network as a <u>web service</u> So all the 10 different application can access it from the network.</p> <p>Interoperability</p> <p>It provides the freedom for a developers to choose whatever the technology they want to use for development.</p> <p>Web services uses a set of standards and protocols and enable us to achieve interoperability.</p> <p>Hence applications developed in Java, Mainframe, Ruby or any other technology can call the <u>web service</u> and use it.</p> <p>Loosely coupled</p> <p><u>Web service</u> exist independent of the other parts of the application that uses it. So any changes to the application can be made without affecting the web service.</p> <p>Deployability</p> <p>It is very easy to deploy the web application as they are deployed over standard internet technologies.</p>
11.	Give the uses of WSDL along with its definition.

WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

Features of WSDL

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

WSDL Usage

WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

12.	<p>Compare SOAP and HTTP.</p> <p>Difference between SOAP and HTTP</p> <table border="1" data-bbox="314 1358 1357 2021"><thead><tr><th data-bbox="314 1358 865 1426">SOAP</th><th data-bbox="865 1358 1357 1426">HTTP</th></tr></thead><tbody><tr><td data-bbox="314 1426 865 2021"><ul style="list-style-type: none">➤ SOAP was originally defined as S- Simple O- Object A-Access P-protocol.➤ It is a protocol specification which is used for exchanging structured information.➤ It is used in the implementation of web services in computer-based networks.➤ SOAP for its message format relies on XML Information set and sometimes relies on other application layer protocols as well, such as Hypertext Transfer Protocol (HTTP) or Simple</td><td data-bbox="865 1426 1357 2021"><ul style="list-style-type: none">➤ The HTTP or Hypertext Transfer Protocol (HTTP) is an application protocol which is used for distributed, collaborative and hypermedia information systems.➤ HTTP is widely regarded as the foundation of data communication for the World Wide Web (WWW).➤ Hypertext is a structured text that uses logical links or hyperlinks between those nodes that contain text. HTTP is the protocol</td></tr></tbody></table>	SOAP	HTTP	<ul style="list-style-type: none">➤ SOAP was originally defined as S- Simple O- Object A-Access P-protocol.➤ It is a protocol specification which is used for exchanging structured information.➤ It is used in the implementation of web services in computer-based networks.➤ SOAP for its message format relies on XML Information set and sometimes relies on other application layer protocols as well, such as Hypertext Transfer Protocol (HTTP) or Simple	<ul style="list-style-type: none">➤ The HTTP or Hypertext Transfer Protocol (HTTP) is an application protocol which is used for distributed, collaborative and hypermedia information systems.➤ HTTP is widely regarded as the foundation of data communication for the World Wide Web (WWW).➤ Hypertext is a structured text that uses logical links or hyperlinks between those nodes that contain text. HTTP is the protocol
SOAP	HTTP				
<ul style="list-style-type: none">➤ SOAP was originally defined as S- Simple O- Object A-Access P-protocol.➤ It is a protocol specification which is used for exchanging structured information.➤ It is used in the implementation of web services in computer-based networks.➤ SOAP for its message format relies on XML Information set and sometimes relies on other application layer protocols as well, such as Hypertext Transfer Protocol (HTTP) or Simple	<ul style="list-style-type: none">➤ The HTTP or Hypertext Transfer Protocol (HTTP) is an application protocol which is used for distributed, collaborative and hypermedia information systems.➤ HTTP is widely regarded as the foundation of data communication for the World Wide Web (WWW).➤ Hypertext is a structured text that uses logical links or hyperlinks between those nodes that contain text. HTTP is the protocol				

	<p>Mail Transfer Protocol (SMTP).</p> <ul style="list-style-type: none"> ➤ It is used for message negotiation and transmission mainly. ➤ SOAP forms the foundation layer of a web services protocol stack. 	<p>for exchanging or transferring hypertext.</p> <ul style="list-style-type: none"> ➤ The standards development of HTTP when it was innovated was coordinated by the Internet Engineering Task Force and the World Wide Web Consortium also called as W3C. 	
13.	<p>Summarize the need for enhancing security in web services.</p> <p>Definition - What does <i>Web Services Security (WS Security)</i> mean?</p> <p>Web Services Security (WS Security) is a specification that defines how security measures are implemented in web services to protect them from external attacks. It is a set of protocols that ensure security for SOAP-based messages by implementing the principles of confidentiality, integrity and authentication.</p> <p>Because Web services are independent of any hardware and software implementations, WS-Security protocols need to be flexible enough to accommodate new security mechanisms and provide alternative mechanisms if an approach is not suitable. Because SOAP-based messages traverse multiple intermediaries, security protocols need to be able to identify fake nodes and prevent data interpretation at any nodes. WS-Security combines the best approaches to tackle different security problems by allowing the developer to customize a particular security solution for a part of the problem. For example, the developer can select digital signatures for non-repudiation and Kerberos for authentication.</p>		
14.	<p>Name the types of indicators along with the definition.</p> <h3 style="color: red;">Web Services Security (WSS)</h3> <p>Web Services Security (WSS or WS-Security) describes enhancements to SOAP messaging in order to provide quality of protection through message integrity, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies.</p> <p>The scope of the Web Services Security Technical Committee is the support of security mechanisms in the following areas:</p>		

	<p>Using XML Signature to provide SOAP message integrity for Web Services</p> <p>Using XML Encryption to provide SOAP message confidentiality for Web Services</p> <p>Attaching and/or referencing security tokens in headers of SOAP messages. Options include:</p> <p>Username token</p> <p>SAML</p> <p>XrML</p> <p>Kerberos</p> <p>X.509</p> <p>Carrying security information for potentially multiple, designated actors</p> <p>Associating signatures with security tokens</p> <p>Each of the security mechanisms will use implementation and language neutral XML formats defined in XML Schema.</p>
	<p>Classify the basic concepts behind JAX-RPC technology.</p> <p>JAX-RPC</p> <p>Java APIs for XML-based Remote Procedure Call (JAX-RPC) help with Web service interoperability and accessibility by defining Java APIs that Java applications use to develop and access Web services. JAX-RPC fully embraces the heterogeneous nature of Web services -- it allows a JAX-RPC client to talk to another Web service deployed on a different platform and coded in a different language. Similarly, it also allows clients on other platforms and coded in different languages to talk to a JAX-RPC service. JAX-RPC also defines the mapping between WSDL service descriptions and Java interfaces.</p>
15.	<p>Th the JAX-RPC technology and describes its client and server programming models. JAX-RPC hides the complexity of underlying protocols and message-level processing from application developers crafting Web services using the Java 2 platform. The API combines XML with Remote Procedure Call (RPC), which is a mechanism enabling clients to execute procedures on distributed or remote systems, so that developers can build Web services and clients. The JAX-RPC remote procedure calls are represented by an XML infoset and they are carried over a network transport. While the JAX-RPC APIs rely on a XML-based protocol and a network transport, the APIs themselves are independent of a specific protocol or transport. The current JAX-RPC implementation relies on the SOAP 1.1 protocol and HTTP 1.1 network transport.</p>
16.	<p>What are the benefits of UDDI?</p> <p>Problems the UDDI specification can help to solve:</p> <ul style="list-style-type: none"> • Making it possible to discover the right business from the millions currently online • Defining how to enable commerce once the preferred business is discovered • Reaching new customers and increasing access to current customers • Expanding offerings and extending market reach

	<ul style="list-style-type: none"> • Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy • Describing services and business processes programmatically in a single, open, and secure environment
17.	<p>What are the core elements of UDDI?</p> <p>UDDI defines four core data elements within the data model:</p> <ul style="list-style-type: none"> • businessEntity (modeling business information) • businessService (describing a service) • tModel (describing specifications, classifications, or identifications) • binding Template (mapping between a businessService and the set of tModels that describe its technical fingerprint)
18.	<p>Rewrite the definition for UDDI.</p> <p>UDDI is an XML-based standard for describing, publishing, and finding web services.</p> <ul style="list-style-type: none"> • UDDI stands for Universal Description, Discovery, and Integration. • UDDI is a specification for a distributed registry of web services. • UDDI is a platform-independent, open framework. • UDDI can communicate via SOAP, CORBA, Java RMI Protocol. • UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services. • UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services. • UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet. <p>UDDI has two sections –</p> <ul style="list-style-type: none"> • A registry of all web service's metadata, including a pointer to the WSDL description of a service. • A set of WSDL port type definitions for manipulating and searching that registry.
19.	<p>Give the usage of UDDI in web service.</p> <p>UDDI is an XML-based standard for describing, publishing, and finding web services.</p> <ul style="list-style-type: none"> • UDDI stands for Universal Description, Discovery, and Integration. • UDDI is a specification for a distributed registry of web services. • UDDI is a platform-independent, open framework.

	<ul style="list-style-type: none"> • UDDI can communicate via SOAP, CORBA, Java RMI Protocol. • UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services. • UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services. • UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet. <p>UDDI has two sections –</p> <ul style="list-style-type: none"> • A registry of all web service's metadata, including a pointer to the WSDL description of a service. • A set of WSDL port type definitions for manipulating and searching that registry.
20.	<p>Define WSDL.</p> <p>WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.</p> <h3>Features of WSDL</h3> <ul style="list-style-type: none"> • WSDL is an XML-based protocol for information exchange in decentralized and distributed environments. • WSDL definitions describe how to access a web service and what operations it will perform. • WSDL is a language for describing how to interface with XML-based services. • WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry. • WSDL is the language that UDDI uses. • WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'. <h3>WSDL Usage</h3> <p>WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.</p>
	PART-B
1.	(i) Describe in detail about the AJAX architecture.

	<p>(ii) List out the call back methods.</p>
	<p>(i) Analyze various concepts of RPC.</p> <h2>Remote Procedure Call (RPC)</h2> <p>A remote procedure call is an interprocess communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call.</p> <p>A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumes execution after the server is finished.</p> <p>The sequence of events in a remote procedure call are given as follows:</p> <ul style="list-style-type: none"> • The client stub is called by the client. • The client stub makes a system call to send the message to the server and puts the parameters in the message. • The message is sent from the client to the server by the client's operating system. • The message is passed to the server stub by the server operating system. • The parameters are removed from the message by the server stub. • Then, the server procedure is called by the server stub. <p>2. A diagram that demonstrates this is as follows:</p> <pre> graph LR subgraph Client [Client] direction TB CF[Client Functions] --> CS[Client Stub] CS --> RR[RPC Runtime] RR --> SF[Server Functions] SF --> SS[Server Stub] SS --> RR end subgraph Server [Server] direction TB SF[Server Functions] --> SS[Server Stub] SS --> RR[RPC Runtime] RR --> CF[Client Functions] end </pre> <p>The diagram illustrates the RPC architecture between a Client and a Server. Both sides have a layered structure. On the Client side, there are 'Client Functions' at the top, followed by the 'Client Stub', and finally the 'RPC Runtime' at the bottom. On the Server side, there are 'Server Functions' at the top, followed by the 'Server Stub', and finally the 'RPC Runtime' at the bottom. Arrows indicate the flow of data: from Client Functions to Client Stub, from Client Stub to RPC Runtime, from RPC Runtime to Server Functions, and from Server Functions to Server Stub. Additionally, there are bidirectional arrows between the RPC Runtimes of the Client and Server, indicating the exchange of messages between the two systems.</p>

Advantages of Remote Procedure Call

Some of the advantages of RPC are as follows:

- Remote procedure calls support process oriented and thread oriented models.
- The internal message passing mechanism of RPC is hidden from the user.
- The effort to re-write and re-develop the code is minimum in remote procedure calls.
- Remote procedure calls can be used in distributed environment as well as the local environment.
- Many of the protocol layers are omitted by RPC to improve performance.

Disadvantages of Remote Procedure Call

Some of the disadvantages of RPC are as follows:

- The remote procedure call is a concept that can be implemented in different ways. It is not a standard.
- There is no flexibility in RPC for hardware architecture. It is only interaction based.
- There is an increase in costs because of remote procedure call.

(ii) **Explain** the basic concepts behind JAX-RPC.

JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.

In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. In this release, JAX-RPC relies on SOAP 1.1 and HTTP 1.1.

Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer. On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy.

With JAX-RPC, clients and Web services have a big advantage--the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide

	<p>Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.</p>
	<p>Explain in detail with an example of Java Web Services.</p> <p>With a simple example illustrate the steps to create a java web service. (NOV/DEC 2012)</p> <p>Writing a java web service</p> <p>Currency conversion Service</p> <ul style="list-style-type: none"> ◆ Writing a server for a service using JWSDP 1.3 tools ◆ Application: currency converter ■ Three operations: ● fromDollars ● fromEuros ● fromYen ■ Input: value in specified currency ■ Output: object containing input value and equivalent values in other two currencies <p>Writing server software</p> <ol style="list-style-type: none"> 1. Write service endpoint interface • May need to write additional classes representing data structures 2. Write class implementing the interface 3. Compile classes 4. Create configuration files and run JWSDP tools to create web service 5. Deploy web service to Tomcat <p>service endpoint interface</p> <ul style="list-style-type: none"> ◆ The Web service endpoint interface is used to define the ‘Web services methods’. ◆ A Web service endpoint interface must conform to the rules of a JAX-RPC service definition interface. ◆ a service endpoint interface (SEI) that defines the interface of the web service. ◆ Configuration files are XML files that can be changed as needed. Developers can use configuration files to change settings without recompiling applications. <p>Administrators can use configuration files to set policies that affect how applications run on their computers.</p> <ul style="list-style-type: none"> ◆ config.xml : Defines the URL for WSDL file location. Each Web services has a corresponding WSDL (Web service Definition Language) document. <p>JWSDP: Server</p>
	<p>Rules for Service endpoint interface</p> <ul style="list-style-type: none"> ■ Must extend java.rmi.Remote ● declares a set of methods that may be invoked from a remote Java Virtual Machine(JVM) ■ Every method must throw java.rmi.RemoteException

- Parameter/return value data types are restricted
- No public static final declarations (global constants) It must not have constant declarations
 - ◆ **Allowable parameter/return value data types**
 - Java primitives (int, boolean, *etc.*)
 - Primitive wrapper classes (Integer, *etc.*)
 - String, Date, Calendar, BigDecimal, BigInteger
 - java.xml.namespace.QName, java.net.URI
 - Struct: class consisting entirely of public instance variables
 - Array of any of the above
 - ◆ **Struct for currency converter app (data type for return values)**

```

package myCurCon;

public class ExchangeValues {
    public double dollars;
    public double euros;
    public double yen;
}

◆ Service endpoint interface
package myCurCon;

public interface CurCon extends java.rmi.Remote {
    public ExchangeValues fromDollars(double dollars)
        throws java.rmi.RemoteException;
    public ExchangeValues fromEuros(double euros)
        throws java.rmi.RemoteException;
    public ExchangeValues fromYen(double yen)
        throws java.rmi.RemoteException;
}
◆ Three files ExchangeValues.java, CurCon.java and CurConImpl.java written for the web service
◆ Class CurConImpl contains methods implements service endpoint interface, for example:
public ExchangeValues fromDollars(double dollars)
    throws java.rmi.RemoteException
{
    ExchangeValues ev = new ExchangeValues();
    ev.dollars = dollars;
    ev.euros = dollars * dollar2euro;
    ev.yen = dollars * dollar2yen;
    return ev;
}

```

Packaging server software

- ◆ Configuration file input to wscompile to create server

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <service
    name="HistoricCurrencyConverter"
    targetNamespace="http://tempuri.org/wsdl"
    typeNamespace="http://tempuri.org/types"
    packageName="myCurCon">
    <interface name="myCurCon.CurCon" />
  </service>
</configuration>
```

- ◆ Configuration file for web service

```
<?xml version="1.0" encoding="UTF-8"?>
<webServices
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/dd"
  version="1.0"
  targetNamespaceBase="http://tempuri.org/wsdl"
  typeNamespaceBase="http://tempuri.org/types"
  >
```

- ◆ Configuration file for web service

```
  urlPatternBase="/converter">
```

Context path

```
<endpoint
  name="CurrConverter"
  displayName="Currency Converter"
  description=
    "Converts between dollars, euros, and yen."
  interface="myCurCon.CurCon"
  model="/WEB-INF/model.xml.gz"
  implementation="myCurCon.CurConImpl"/>
```

Like servlet in web.xml

```
<endpointMapping
  endpointName="CurrConverter"
  urlPattern="/currency" />
```

Like servlet-mapping in web.xml

```
</webServices>
```

- ◆ Also need a minimal web.xml

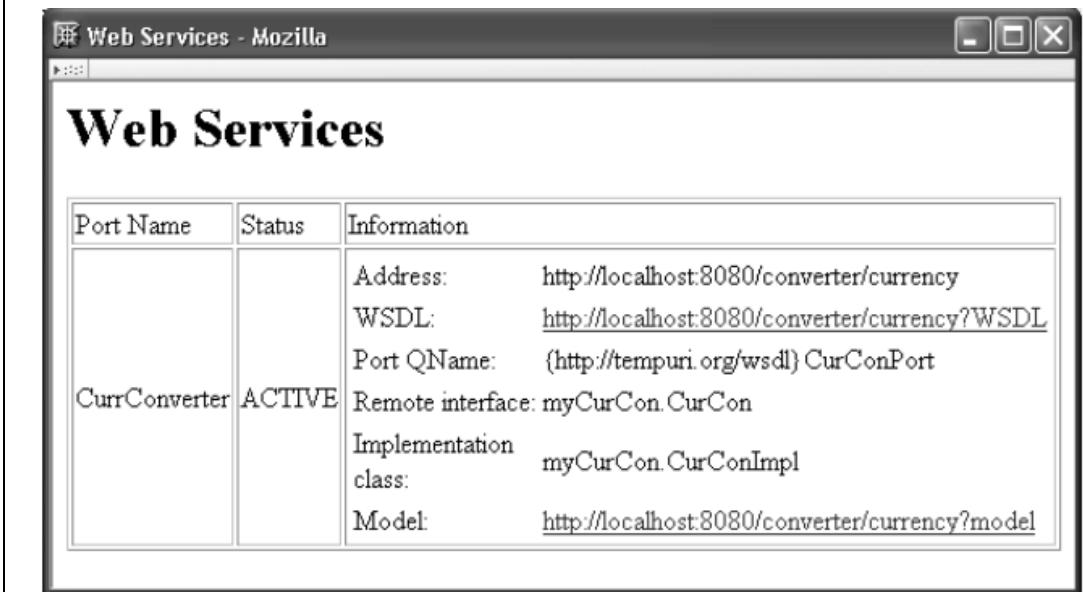
```
<web-app
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">
```

```
  <display-name>Historic Currency Converter</display-name>
  <description>
    This web service converts between three currencies using their
    exchange rates as of a fixed date.
  </description>
</web-app>
```

Run jar and wsdeploy to create a Web Archive (WAR) file converter.war

- Name must match urlPatternBase value

- ◆ `jaxrpc-ri.xml`: Defines the various end points for referencing a Web service.
- ◆ `wscompile`: The wscompile tool generates stubs, and WSDL files used in JAX-RPC clients and services. The tool reads as input a configuration file and either a WSDL file or an RMI interface that defines the service.
- ◆ `wsdeploy`: Reads a WAR file (something like Jar file) and the `jaxrpc-ri.xml` file and then generates another WAR file that is ready for deployment
 - ◆ Write service endpoint interface
 - May need to write additional classes representing data structures
 - ◆ Write class implementing the interface
 - ◆ Compile classes
 - ◆ Create configuration files and run JWSDP tools to create web service
 - ◆ Deploy web service to Tomcat
 - ◆ Just copy converter.war to Tomcat webapps directory
 - May need to use Manager app to deploy
 - Enter converter.war in “WAR or Directory URL” text box
 - ◆ Testing success:
 - Visit <http://localhost:8080/converter/currency>



Discuss in detail the architecture of web services.

Architecture of Web Services

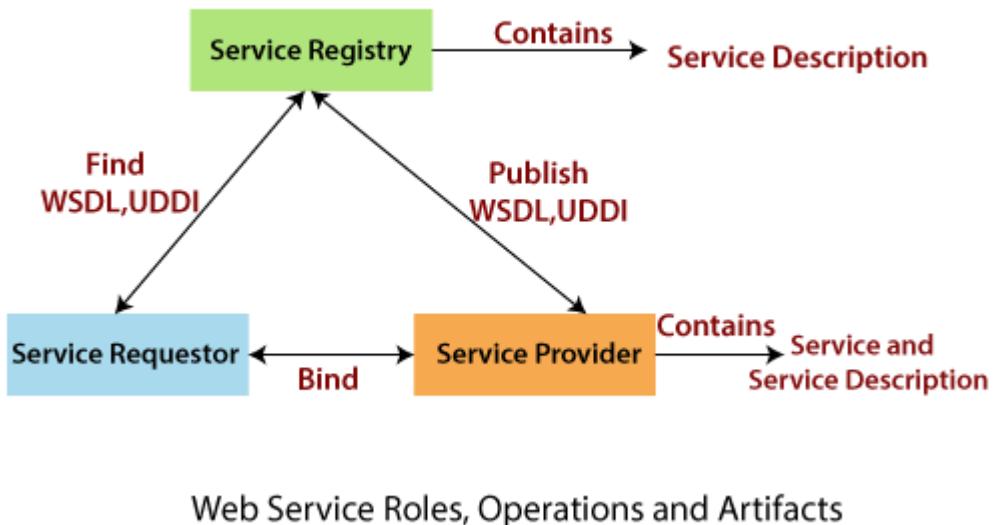
4. The Web Services architecture describes how to instantiate the elements and implement the operations in an interoperable manner.

The architecture of web service interacts among three roles: **service provider**, **service requester**, and **service registry**. The interaction involves the three operations: **publish**, **find**, and **bind**. These operations and roles act upon

the **web services artifacts**. The web service artifacts are the web service software module and its description.

The service provider hosts a network-associable module (web service). It defines a service description for the web service and publishes it to a service requestor or service registry. These service requestor uses a find operation to retrieve the service description locally or from the service registry. It uses the service description to bind with the service provider and invoke with the web service implementation.

The following figure illustrates the operations, roles, and their interaction.



Roles in a Web Service Architecture

There are three roles in web service architecture:

- Service Provider
- Service Requestor
- Service Registry

Service Provider

From an architectural perspective, it is the platform that hosts the services.

Service Requestor

Service requestor is the application that is looking for and invoking or initiating an interaction with a service. The browser plays the requester role, driven by a consumer or a program without a user interface.

Service Registry

Service requestors find service and obtain binding information for services during development.

Operations in a Web Service Architecture

Three behaviors that take place in the microservices:

- Publication of service descriptions (**Publish**)
- Finding of services descriptions (**Find**)
- Invoking of service based on service descriptions (**Bind**)

Publish: In the publish operation, a service description must be published so that a service requester can find the service.

Find: In the find operation, the service requestor retrieves the service description directly. It can be involved in two different lifecycle phases for the service requestor:

- At design, time to retrieve the service's interface description for program development.
- And, at the runtime to retrieve the service's binding and location description for invocation.

Bind: In the bind operation, the service requestor invokes or initiates an interaction with the service at runtime using the binding details in the service description to locate, contact, and invoke the service.

Artifacts of the web service

There are two artifacts of web services:

- Service
- Service Registry

Service: A service is an **interface** described by a service description. The service description is the implementation of the service. A service is a software module deployed on network-accessible platforms provided by the service provider. It interacts with a service requestor. Sometimes it also functions as a requestor, using other Web Services in its implementation.

Service Description: The service description comprises the details of the **interface** and **implementation** of the service. It includes its **data types**, **operations**, **binding information**, and **network location**. It can also categorize other metadata to enable discovery and utilize by service requestors. It can be published to a service requestor or a service registry.

5. (i) Deduce any two elements of WSDL.

WSDL (Web services description language)

A web service cannot be used if it cannot be found. The client invoking the web service should know where the web service actually resides.

Secondly, the client application needs to know what the web service actually does, so that it can invoke the right web service. This is done with the help of the WSDL, known as the Web services description language. The WSDL file is again an XML-based file which basically tells the client application what the web service does. By using the WSDL document, the client application would be able to understand where the web service is located and how it can be utilized.

Web Service Example

An example of a WSDL file is given below.

```
<definitions>
    <message name="TutorialRequest">
        <part name="TutorialID" type="xsd:string"/>
    </message>

    <message name="TutorialResponse">
        <part name="TutorialName" type="xsd:string"/>
    </message>

    <portType name="Tutorial_PortType">
        <operation name="Tutorial">
            <input message="tns:TutorialRequest"/>
            <output message="tns:TutorialResponse"/>
        </operation>
    </portType>

    <binding name="Tutorial_Binding" type="tns:Tutorial_PortType">
        <soap:binding style="rpc"
                      transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="Tutorial">
            <soap:operation soapAction="Tutorial"/>
            <input>
                <soap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/>
                    namespace="urn:examples:Tutorialservice"
```

```

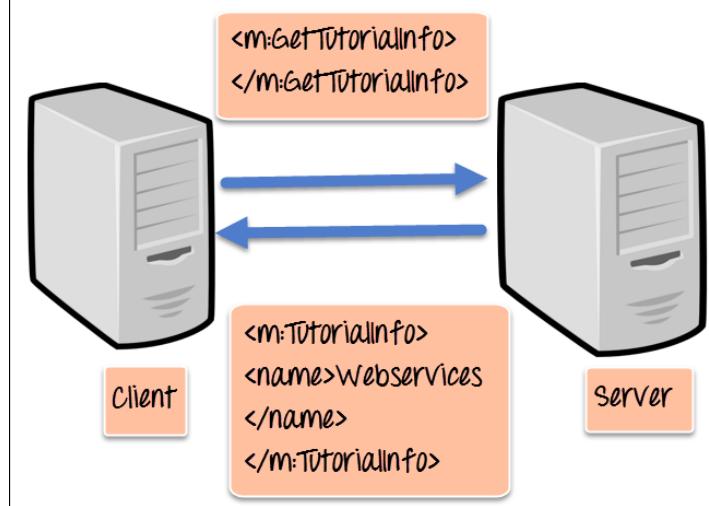
        use="encoded"/>
    </input>

    <output>
        <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:Tutorialservice"
            use="encoded"/>
    </output>
</operation>
</binding>
</definitions>
```

(ii) **Explain** the steps for writing web service.

6.	<p>Describe briefly about the elements of WSDL.</p> <p>The important aspects to note about the above WSDL declaration are as follows;</p> <ol style="list-style-type: none"> 1. <message> - The message parameter in the WSDL definition is used to define the different data elements for each operation performed by the web service. So in the example above, we have 2 messages which can be exchanged between the web service and the client application, one is the "TutorialRequest", and the other is the "TutorialResponse" operation. The TutorialRequest contains an element called "TutorialID" which is of the type string. Similarly, the TutorialResponse operation contains an element called "TutorialName" which is also a type string. 2. <portType> - This actually describes the operation which can be performed by the web service, which in our case is called Tutorial. This operation can take 2 messages; one is an input message, and the other is the output message. 3. <binding> - This element contains the protocol which is used. So in our case, we are defining it to use http (http://schemas.xmlsoap.org/soap/http). We also specify other details for the body of the operation, like the
----	--

	<p>namespace and whether the message should be encoded.</p>
7.	<p>(i) Summarize on the structure of SOAP.</p> <h3 style="text-align: center;">SOAP (Simple Object Access Protocol)</h3> <p>SOAP is known as a transport-independent messaging protocol. SOAP is based on transferring XML data as SOAP Messages. Each message has something which is known as an XML document. Only the structure of the XML document follows a specific pattern, but not the content. The best part of Web services and SOAP is that its all sent via HTTP, which is the standard web protocol.</p> <p>Here is what a SOAP message consists of</p> <ul style="list-style-type: none"> ○ Each SOAP document needs to have a root element known as the <Envelope> element. The root element is the first element in an XML document. ○ The "envelope" is in turn divided into 2 parts. The first is the header, and the next is the body. ○ The header contains the routing data which is basically the information which tells the XML document to which client it needs to be sent to. ○ The body will contain the actual message. <p>The diagram below shows a simple example of the communication via SOAP.</p>



(ii) **Describe** briefly about SOAP & HTTP.

(i) **Demonstrate** the building blocks of SOAP.

XML Soap

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation

8. Why SOAP?

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

(ii) **Classify** the encoding of struct data and array.

SOAP - Encoding

SOAP includes a built-in set of rules for encoding data types. It enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

- SOAP data types are divided into two broad categories – scalar types and compound types.
- Scalar types contain exactly one value such as a last name, price, or product description.
- Compound types contain multiple values such as a purchase order or a list of stock quotes.
- Compound types are further subdivided into arrays and structs.

Compound Types

SOAP arrays have a very specific set of rules, which require that you specify both the element type and array size. SOAP also supports multidimensional arrays, but not all SOAP implementations support multidimensional functionality.

To create an array, you must specify it as an `xsi:type` of array. The array must also include an `arrayType` attribute. This attribute is required to specify the data type for the contained elements and the dimension(s) of the array.

For example, the following attribute specifies an array of 10 double values –

```
arrayType = "xsd:double[10]"
```

In contrast, the following attribute specifies a two-dimensional array of strings –

```
arrayType = "xsd:string[5,5]"
```

Here is a sample SOAP response with an array of double values –

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
    envelope"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:getPriceListResponse
            xmlns:ns1 = "urn:examples:pricelistservice"
            SOAP-ENV:encodingStyle =
            "http://www.w3.org/2001/12/soap-encoding">

            <return xmlns:ns2 =
            "http://www.w3.org/2001/09/soap-encoding"
            xsi:type = "ns2:Array" ns2:arrayType =
            "xsd:double[2]">
                <item xsi:type = "xsd:double">54.99</item>
                <item xsi:type = "xsd:double">19.99</item>
            </return>
        </ns1:getPriceListResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Structs contain multiple values, but each element is specified with a unique accessor element. For example, consider an item within a product catalog. In this case, the struct might contain a product SKU, product name, description, and price. Here is how such a struct would be represented in a SOAP message –

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
    envelope"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:getProductResponse
            xmlns:ns1 = "urn:examples:productservice">
```

```

SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">

    <return xmlns:ns2 = "urn:examples" xsi:type =
"ns2:product">
        <name xsi:type = "xsd:string">Red Hat
Linux</name>
        <price xsi:type = "xsd:double">54.99</price>
        <description xsi:type = "xsd:string">
            Red Hat Linux Operating System
        </description>
        <SKU xsi:type = "xsd:string">A358185</SKU>
    </return>
</ns1:getProductResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Analyze the various steps in database driven web service with some example.

Overview of Database Web Services

Web services enable application-to-application interaction over the Web, regardless of platform, language, or data formats. The key ingredients, including Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI), have been adopted across the entire software industry. Web services usually refer to services implemented and deployed in middle-tier application servers. However, in heterogeneous and disconnected environments, there is an increasing need to access stored procedures, as well as data and metadata, through Web services interfaces.

The Database Web services technology is a database approach to Web services. It works in the following two directions:

9.

- Accessing database resources as a Web service
- Consuming external Web services from the database

Oracle Database can access Web services through PL/SQL packages and Java classes deployed within the database. Turning Oracle Database into a Web service provider leverages investment in Java stored procedures, PL/SQL packages, predefined SQL queries, and data manipulation language (DML). Conversely, consuming external Web services from the database, together with integration with the SQL engine, enables Enterprise Information Integration.

Using Oracle Database as Web Services Provider

Web Services use industry-standard mechanisms to provide easy access to remote content and applications, regardless of the platform and location of the provider and implementation and data format. Client applications can query and retrieve data from Oracle Database and call stored procedures using standard Web service protocols. There is no dependency on Oracle-specific

	<p>database connectivity protocols. This approach is highly beneficial in heterogeneous, distributed, and disconnected environments.</p> <p>You can call into the database from a Web service, using the database as a service provider. This enables you to leverage existing or new SQL, PL/SQL, Java stored procedures, or Java classes within Oracle Database. You can access and manipulate database tables from a Web service client.</p>
	<p>Illustrate on web services for writing web service client along with the description of WSDL.</p> <h2>Creating a Web Service Client</h2> <p>Creating a web service client application always starts with an existing WSDL file.</p> <p>Typically, you retrieve the WSDL directly from a web service provider using the <code>wsimport</code> tool. The <code>wsimport</code> tool then generates the corresponding Java source code for the interface described by the WSDL. The Java compiler, <code>javac</code>, is then called to compile the source into class files. The programming code uses the generated classes to access the web service.</p> <h3>Creating a Client from WSDL</h3> <p>To create a client from WSDL, you must create the following files:</p> <ul style="list-style-type: none"> • Client Java File (fromwsdl) • Client Configuration File (fromwsdl) • <code>build.xml</code> • <code>build.properties</code> <p>Client Java File (fromwsdl)</p> <p>The client Java file defines the functionality of the web service client. The following code shows the <code>AddNumbersClient.java</code> file that is provided in the sample.</p> <pre> package fromjava.client; import com.sun.xml.ws.Closeable; import java.rmi.RemoteException; public class AddNumbersClient { public static void main (String[] args) { AddNumbersImpl port = null; try { port = new AddNumbersImplService().getAddNumbersImplPort(); int number1 = 10; int number2 = 20; System.out.printf ("Invoking addNumbers(%d, %d)\n", number1, number2); int result = port.addNumbers (number1, number2); System.out.printf ("The result of adding %d and %d is %d.\n\n", number1, number2, result); number1 = -10; } } </pre>
10.	

```

        System.out.printf ("Invoking addNumbers(%d, %d)\n",
                           number1, number2);
        result = port.addNumbers (number1, number2);
        System.out.printf (
            "The result of adding %d and %d is %d.\n",
            number1, number2, result);
    } catch (AddNumbersException_Exception ex) {
        System.out.printf (
            "Caught AddNumbersException_Exception: %s\n",
            ex.getFaultInfo ().getDetail ());
    } finally {
        ((Closeable)port).close();
    }
}
}

```

This file specifies two positive integers that are to be added by the web service, passes the integers to the web service and gets the results from the web service via the `port.addNumbers` method, and prints the results to the screen. It then specifies a negative number to be added, gets the results (which should be an exception), and prints the results (the exception) to the screen.

Client Configuration File (fromwsdl)

This is a sample `custom-client.xml` file. The `wsdlLocation`, package name, and `jaxb:package` name `xml` tags are unique to each client and are highlighted in bold text

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bindings
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    wsdlLocation="http://localhost:8080/wsit-enabled-fromwsdl/
        addnumbers?wsdl"
    xmlns="http://java.sun.com/xml/ns/jaxws">
    <bindings node="ns1:definitions"
        xmlns:ns1="http://schemas.xmlsoap.org/wsdl/">
        <package name="fromwsdl.client" />
    </bindings>
    <bindings node="ns1:definitions/ns1:types/xsd:schema
        [@targetNamespace='http://duke.org']"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:ns1="http://schemas.xmlsoap.org/wsdl/">
        <jaxb:schemaBindings>
            <jaxb:package name="fromwsdl.client" />
        </jaxb:schemaBindings>
    </bindings>
</bindings>

```

- | | |
|--|---|
| | <p>(i) List out the installation steps of JWSDP.</p> <p>JWSDP</p> <p>11. 1. Install JDK 6.0 (i.e., JDK 1.6.0)</p> <p>Set up the following environment variables:
 <code>JAVA_HOME C:\Program Files\Java\jdk1.6.0_07</code></p> <p>Add the following path:
 <code>C:\Program Files\Java\jdk1.6.0_07\bin</code></p> |
|--|---|

2. Install [JWSDP 2.0](#) & [Tomcat 5.0 for JWSDP](#) (based upon Tomcat 5.0.19 that implements the Java Server Pages 2.0 and Java Servlet 2.4 specifications)

Set up the following environment variables:
 JWSDP_HOME C:\Sun\jwsdp-2.0
 ANT_HOME C:\Sun\jwsdp-2.0\apache-ant

Add the following path:
 C:\Sun\jwsdp-2.0\jwsdp-shared\bin;C:\Sun\jwsdp-2.0\apache-ant\bin

3. Copy [examples.zip](#) into C:\ and extract here

4. Copy [lib.zip](#) into C:\Sun\jwsdp-2.0\server directory and extract here
 Delete the file "lib.zip"

5. Replace saaj-impl.jar file at the following directories by [saaj-impl-1.3.jar](#).
 Rename it to saaj-impl.jar.

C:\Sun\jwsdp-2.0\saaj\lib
 C:\Sun\tomcat50-jwsdp\saaj\lib

6. Modify C:\examples\common\build.properties for the first four lines as follows:

```
tutorial.home=C:
tutorial.install=${tutorial.home}
username=hxu
password=12345
```

where "hxu" and "12345" are the username and password for the Tomcat server.

7. Build server:
 cd C:\examples\jaxrpc\helloservice
 ant build

Start Tomcat from JWSDP 2.0

Deploy server:
 ant deploy

Note: If application already exists at path /hello-jaxrpc, you should use the command "ant undeploy" to undeploy the web service first.

Verify the deployment:
 To verify that the service has been successfully deployed, open a browser window and specify the service endpoint's URL as follows:

<http://localhost:8080/hello-jaxrpc/hello?WSDL>

You should get the following display.

```

<?xml version="1.0" encoding="UTF-8"?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="MyHelloService" targetNamespace="urn:Foo">
  <types />
  - <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string" />
  </message>
  + <message name="HelloIF_sayHelloResponse">
  - <portType name="HelloIF">
    - <operation name="sayHello" parameterOrder="String_1">
      <input message="tns:HelloIF_sayHello" />
      <output message="tns:HelloIF_sayHelloResponse" />
    </operation>
  </portType>
  - <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
    - <operation name="sayHello">
      <soap:operation soapAction="" />
      - <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="urn:Foo" />
      </input>
      - <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="urn:Foo" />
      </output>
    </operation>
  </binding>
  - <service name="MyHelloService">
    - <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address location="http://localhost:8080/hello-jaxrpc/hello" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
    </port>
  </service>
</definitions>

```

8. Build client:

```
cd C:\examples\jaxrpc\dynamicproxy
ant build
```

Run client:
ant run

```

C:\ Command Prompt
[echo] Creating the required directories.....
package-dynamic:
[echo] Building the client JAR file....
[delete] Deleting: C:\examples\jaxrpc\dynamicproxy\dist\client.jar
[jar] Building jar: C:\examples\jaxrpc\dynamicproxy\dist\client.jar

build-dynamic:

build:

BUILD SUCCESSFUL
Total time: 20 seconds
C:\examples\jaxrpc\dynamicproxy>ant run
Buildfile: build.xml

run-client:
[javal UrlString = http://localhost:8080/hello-jaxrpc/hello?WSDL
[javal Hello Buzz

run:

BUILD SUCCESSFUL
Total time: 3 seconds
C:\examples\jaxrpc\dynamicproxy>

```

(ii) **Describe** on Simple Object Access Protocol.

XML Soap

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation

Why SOAP?

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

12.	(i) Discuss the XMLHttpRequest Object with example.
-----	--

AJAX - The XMLHttpRequest Object

The keystone of AJAX is the XMLHttpRequest object.

The XMLHttpRequest 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 web page, without reloading the whole page.

Create an 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();
```

Example

```
var xhttp = new XMLHttpRequest();
```

Access Across Domains

For security reasons, modern browsers do not allow access across domains.

This means that both the web page and the XML file it tries to load, must be located on the same server.

The examples on W3Schools all open XML files located on the W3Schools domain.

If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.

Example

```
if (window.XMLHttpRequest) {  
    // code for modern browsers  
    xmlhttp = new XMLHttpRequest();  
} else {  
    // code for old IE browsers
```

```
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

(ii) **Describe** about Java web service basics.

JAX-WS Example RPC Style

Creating JAX-WS example is a easy task because it requires no extra configuration settings.

JAX-WS API is inbuilt in JDK, so you don't need to load any extra jar file for it. Let's see a simple example of JAX-WS example in RPC style.

There are created 4 files for hello world JAX-WS example:

1. HelloWorld.java
2. HelloWorldImpl.java
3. Publisher.java
4. HelloWorldClient.java

The first 3 files are created for server side and 1 application for client side.

JAX-WS Server Code

File: HelloWorld.java

```
package com.javatpoint;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;
//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface HelloWorld{
    @WebMethod String getHelloWorldAsString(String name);
}
```

File: HelloWorldImpl.java

```
package com.javatpoint;
import javax.jws.WebService;
//Service Implementation
@WebService(endpointInterface = "com.javatpoint.HelloWorld")
```

```
public class HelloWorldImpl implements HelloWorld{
    @Override
    public String getHelloWorldAsString(String name) {
        return "Hello World JAX-WS " + name;
    }
}
```

File: Publisher.java

```
package com.javatpoint;
import javax.xml.ws.Endpoint;
//Endpoint publisher
public class HelloWorldPublisher{
    public static void main(String[] args) {
        Endpoint.publish("http://localhost:7779/ws/hello", new HelloWorldIm
pl());
    }
}
```

How to view generated WSDL

After running the publisher code, you can see the generated WSDL file by visiting the URL:

<http://localhost:7779/ws/hello?wsdl>

JAX-WS Client Code

File: HelloWorldClient.java

```
package com.javatpoint;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
public class HelloWorldClient{
    public static void main(String[] args) throws Exception {
        URL url = new URL("http://localhost:7779/ws/hello?wsdl");

        //1st argument service URI, refer to wsdl document above
        //2nd argument is service name, refer to wsdl document above
        QName qname = new QName("http://javatpoint.com/", "HelloWorld
ImplService");
        Service service = Service.create(url, qname);
        HelloWorld hello = service.getPort(HelloWorld.class);
```

```

.     System.out.println(hello.getHelloWorldAsString("javatpoint rpc"));
.   }
. }

}

```

Output:

```
Hello World JAX-WS javatpoint rpc
```

	<p>(i) Explain in detail about SOAP encoding.</p> <ul style="list-style-type: none"> For transfer between client and server in a SOAP message, we encode them in XML. <p>SOAP Encoding is an extension of the SOAP framework specification that defines how a data value should be encoded in an XML format. SOAP Data Model is defined as an adjunct in SOAP 1.2 specification.</p> <p>SOAP encoding offers the following rules to convert any data value defined in SOAP data model into XML format. Converting a data value into XML format is called serialization or encoding.</p> <p>Rule 1. A simple value node with a labeled inbound edge will be serialized into a single XML element with the edge's label as the element's name and node value as the element's text content.</p> <p>Rule 2. When serializing a node into an XML element, an "xsi:type" attribute can be added to specify the value type of this note. For more information on "xsi:type", see the other sections in this book.</p> <p>Rule 3. A compound value node with labeled outbound edges, a data structure, will be serialized into a single XML element with child elements. One outbound edge will be serialized into one child element with element's name equal to the edge's label. The order of child elements is not significant.</p> <p>Rule 4. A compound value node with non-labeled outbound edges, a data array, will be serialized into a single XML element with child elements. One outbound edge will be serialized into one child element with element's name equal to any label as long as it's the same for all child elements. The order of child elements signifies the position values of outbound edges.</p> <p>Rule 5. When serializing an array, an "enc:itemType" attribute can be added to specify the value type of its sub nodes, and an "enc:arraySize" attribute can be added to specify the number of values in the array.</p> <p>(ii) Point out the RPC representation model.</p> <h2>What Is JAX-RPC?</h2> <p>JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.</p>
13.	

In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. In this release, JAX-RPC relies on SOAP 1.1 and HTTP 1.1.

Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer. On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy.

With JAX-RPC, clients and Web services have a big advantage--the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.

- Explain** the structure of a WSDL document, its elements and their purposes with appropriate examples.
- A WSDL document defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** which are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitutes a reusable **binding**. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:
- **Types**— a container for data type definitions using some type system (such as XSD).
 - **Message**— an abstract, typed definition of the data being communicated.

- **Operation**— an abstract description of an action supported by the service.
- **Port Type**—an abstract set of operations supported by one or more endpoints.
- **Binding**— a concrete protocol and data format specification for a particular port type.
- **Port**— a single endpoint defined as a combination of a binding and a network address.
- **Service**— a collection of related endpoints.

In addition, WSDL defines a common **binding** mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions.

WSDL Document Example

The following example shows the WSDL definition of a simple service providing stock quotes. The service supports a single operation called GetLastTradePrice, which is deployed using the SOAP 1.1 protocol over HTTP. The request takes a ticker symbol of type string, and returns the price as a float. A detailed description of the elements used in this definition can be found in Section 2 (core language) and Section 3 (SOAP binding).

This example uses a fixed XML format instead of the SOAP encoding (for an example using the SOAP encoding, see [Example 4](#)).

Example 1 SOAP 1.1 Request/Response via HTTP

```

<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd1="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>
  <schema targetNamespace="http://example.com/stockquote.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="TradePriceRequest">
      <complexType>
        <all>
          <element name="tickerSymbol" type="string"/>
        </all>
      </complexType>
    </element>
    <element name="TradePrice">
      <complexType>
        <all>
          <element name="price" type="float"/>
        </all>
      </complexType>
    </element>
  </schema>
</types>

```

	<pre> </all> </complexType> </element> </schema> </types> <message name="GetLastTradePriceInput"> <part name="body" element="xsd1:TradePriceRequest"/> </message> <message name="GetLastTradePriceOutput"> <part name="body" element="xsd1:TradePrice"/> </message> <portType name="StockQuotePortType"> <operation name="GetLastTradePrice"> <input message="tns:GetLastTradePriceInput"/> <output message="tns:GetLastTradePriceOutput"/> </operation> </portType> <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType"> <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/> <operation name="GetLastTradePrice"> <soap:operation soapAction="http://example.com/GetLastTradePrice"/> <input> <soap:body use="literal"/> </input> <output> <soap:body use="literal"/> </output> </operation> </binding> <service name="StockQuoteService"> <documentation>My first service</documentation> <port name="StockQuotePort" binding="tns:StockQuoteBinding"> <soap:address location="http://example.com/stockquote"/> </port> </service> </definitions> </pre>
PAR T – C	
Q.No	<p>Questions</p> <p>Create an XML HttpRequest to retrieve data from an XML file and display the data in an HTML table. The data to be retrieved is a collection of stationary items stored in an XML file.</p> <p>1. <u>The XML Document Used</u></p> <p>INPUT: XML file called "cd_catalog.xml".</p>

Display XML Data in an HTML Table

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

Example

```
<html>
<head>
<style>
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
th, td {
    padding: 5px;
}
</style>
</head>
<body>

<table id="demo"></table>

<script>
function loadXMLDoc() {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            myFunction(this);
        }
    };
    xmlhttp.open("GET", "cd_catalog.xml", true);
    xmlhttp.send();
}
function myFunction(xml) {
    var i;
    var xmlDoc = xml.responseXML;
    var table = "<tr><th>Artist</th><th>Title</th></tr>";
    var x = xmlDoc.getElementsByTagName("CD");
    for (i = 0; i < x.length; i++) {
        table += "<tr><td>" +
        x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
        "</td><td>" +
        x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
        "</td></tr>";
    }
}
loadXMLDoc();
```

```

        }
        document.getElementById("demo").innerHTML = table;
    }
</script>

</body>
</html>

```

OUTPUT

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only
Dr.Hook	Sylvias Mother

Summarize Ajax Client server architecture in detail.

What is AJAX?

AJAX = Asynchronous JavaScript And XML.

AJAX is not a programming language.

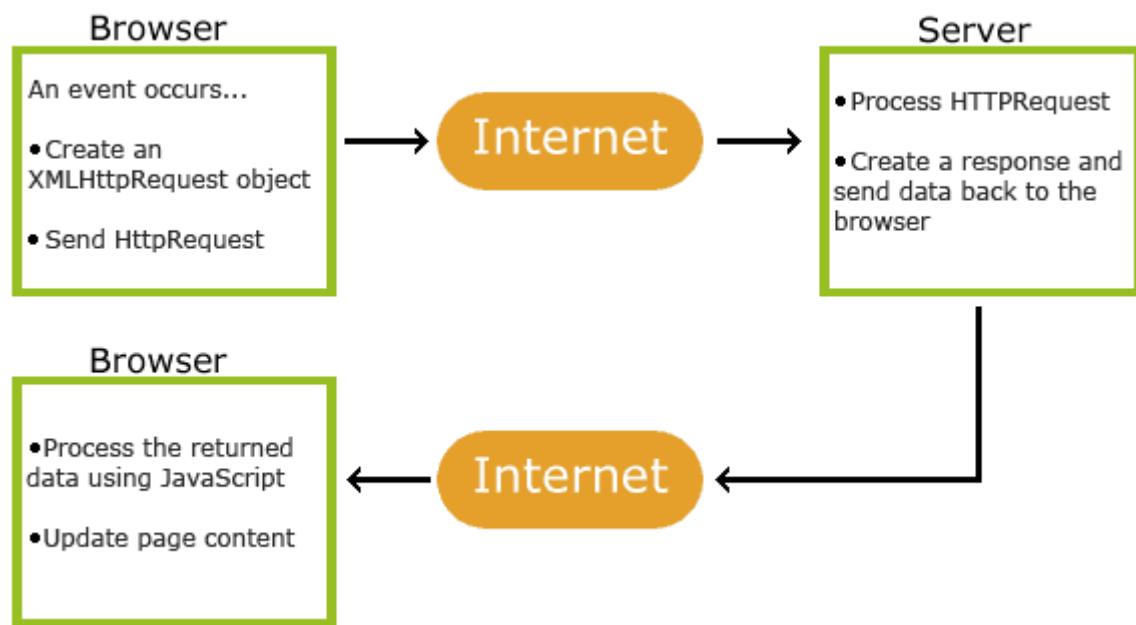
AJAX just uses a combination of:

- 2.
- A browser built-in XMLHttpRequest object (to request data from a web server)
 - JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

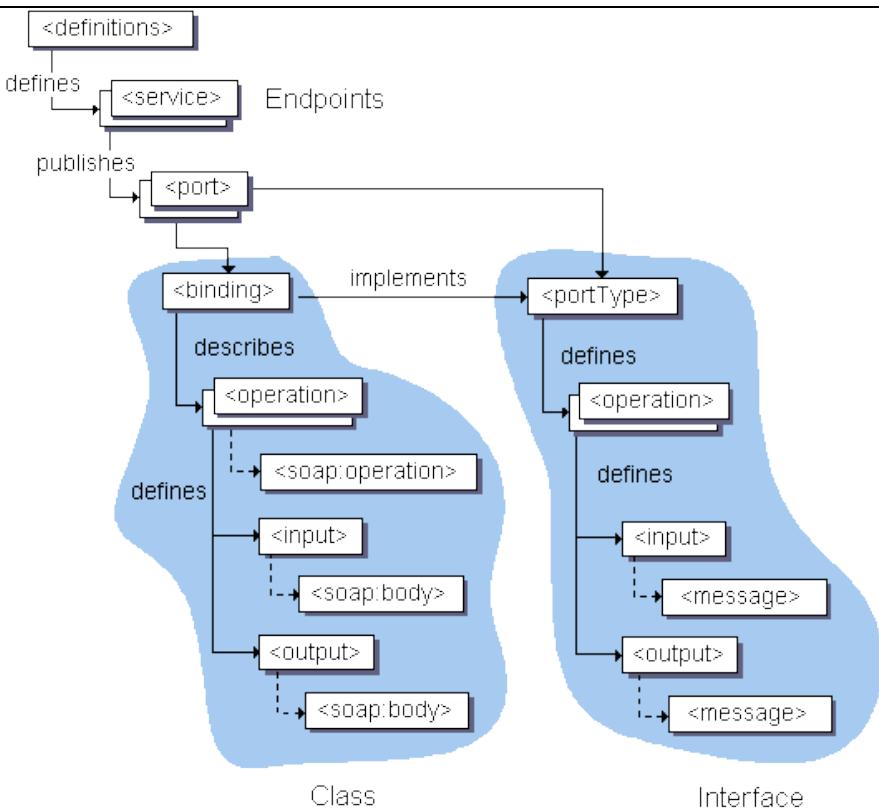
AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

How AJAX Works



1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

	<p>Give the basic structure of a WSDL and show how they are used to create, publish, test and describe web services.</p> <h2>Structure of a WSDL Document</h2> <p>3. Web Services Description Language (WSDL) is an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. The diagram below illustrates the elements that are present in a WSDL document, and indicates their relationships. To see an example of how this is implemented in a WSDL document, see Example of a WSDL Document.</p>
--	--



WSDL Document Elements

A WSDL document has a `definitions` element that contains the other five elements, types, message, portType, binding and service. The following sections describe the features of the generated client code.

WSDL supports the XML Schemas specification (XSD) as its type system.

definitions

Contains the definition of one or more services. JDeveloper generates the following attribute declarations for this section:

- `name` is optional.
- `targetNamespace` is the logical namespace for information about this service. WSDL documents can import other WSDL documents, and setting `targetNamespace` to a unique value ensures that the namespaces do not clash.
- `xmlns` is the default namespace of the WSDL document, and it is set to <http://schemas.xmlsoap.org/wsdl/>.
- All the WSDL elements, such as `<definitions>`, `<types>` and `<message>` reside in this namespace.

	<ul style="list-style-type: none"> • <code>xmlns:xsd</code> and <code>xmlns:soap</code> are standard namespace definitions that are used for specifying SOAP-specific information as well as data types. • <code>xmlns:tns</code> stands for this namespace. • <code>xmlns:ns1</code> is set to the value of the <code>schema targetNamespace</code>, in the <code><types></code> section.
4.	<p>Compare and contrast the additional web application architecture and AJAX Based web application architecture.</p> <p><u>Traditional Web Applications vs. Ajax Applications</u></p> <p>The following highlights the key differences between traditional web applications and Ajax-based web applications.</p> <p><i>Traditional Web Applications</i></p> <ul style="list-style-type: none"> ➤ Figure 15.1 presents the typical interactions between the client and the server in a traditional web application, such as one that uses a user registration form. ➤ First, the user fills in the form's fields, then submits the form (Fig. 15.1, <i>Step 1</i>). The browser generates a request to the server, which receives the request and processes it (<i>Step 2</i>). ➤ The server generates and sends a response containing the exact page that the browser will render (<i>Step 3</i>), which causes the browser to load the new page (<i>Step 4</i>) and temporarily makes the browser window blank. Note that the client <i>waits</i> for the server to respond and <i>reloads the entire page</i> with the data from the response (<i>Step 4</i>). ➤ While such a synchronous request is being processed on the server, the user cannot interact with the client web page.

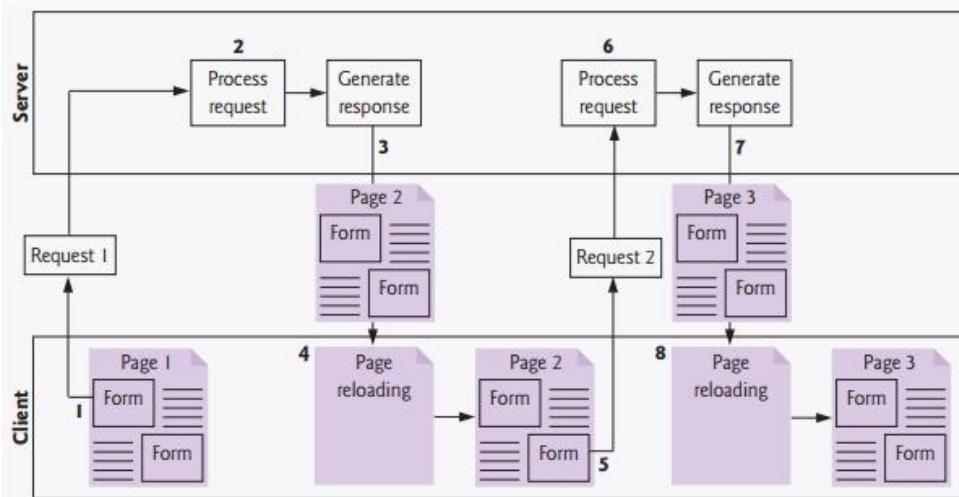


Fig. 15.1 | Classic web application reloading the page for every user interaction.

- Frequent long periods of waiting, due perhaps to Internet congestion, have led some users to refer to the World Wide Web as the “World Wide Wait.”
- If the user interacts with and submits another form, the process begins again (*Steps 5–8*).

This model was originally designed for a web of hypertext documents—what some people call the “brochure web.”

As the web evolved into a full-scale applications platform, the model shown in Fig. 15.1 yielded “choppy” application performance.

Every full-page refresh required users to re-establish their understanding of the full-page contents.

Users began to demand a model that would yield the responsive feel of desktop applications.

Ajax Web Applications

- Ajax applications add a layer between the client and the server to manage communication between the two (Fig. 15.2). When the user interacts with the page, the client creates an XMLHttpRequest object to manage a request (*Step 1*).
- The XMLHttpRequest object sends the request to the server (*Step 2*) and awaits the response.
- The requests are **asynchronous**, so the user can continue interacting with the application on the client-side while the server processes the

earlier request concurrently. Other user interactions could result in additional requests to the server (*Steps 3 and 4*).

- Once the server responds to the original request (*Step 5*), the XMLHttpRequest object that issued the request calls a client-side function to process the data returned by the server.
- This function—known as a **callback function**—uses **partial page updates** (*Step 6*) to display the data in the existing web page *without re-loading the entire page*. At the same time, the server may be responding to the second request (*Step 7*) and the client-side may be starting to do another partial page update (*Step 8*).
- The callback function updates only a designated part of the page.
- Such partial page updates help make web applications more responsive, making them feel more like desktop applications.
- The web application does not load a new page while the user interacts with it.

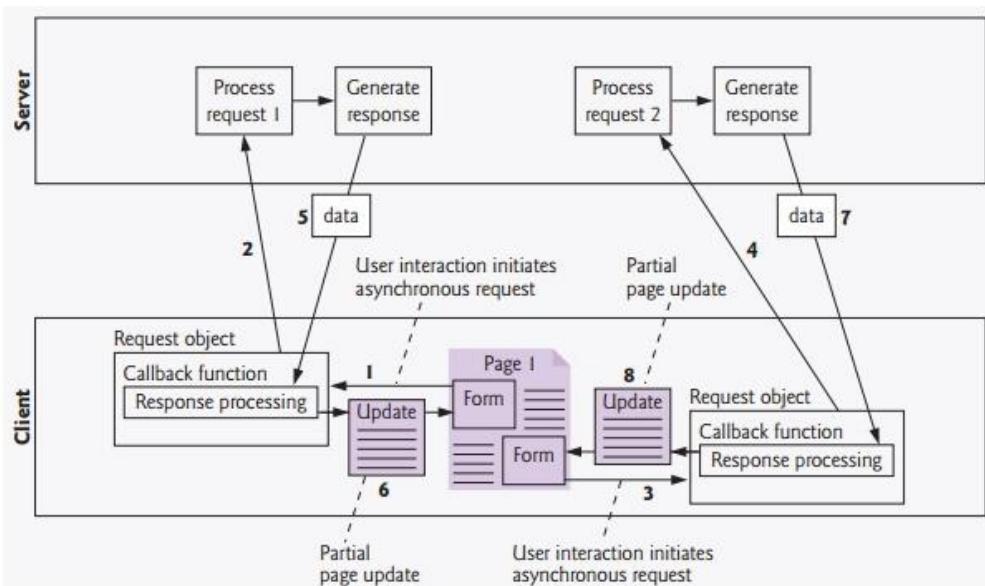


Fig. 15.2 | Ajax-enabled web application interacting with the server asynchronously.

UNIT I WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0**9**

Web Essentials: Clients, Servers and Communication – The Internet – Basic Internet protocols – World wide web – HTTP Request Message – HTTP Response Message – Web Clients – Web Servers – HTML5 – Tables – Lists – Image – HTML5 control elements – Semantic elements – Drag and Drop – Audio – Video controls – CSS3 – Inline, embedded and external style sheets – Rule cascading – Inheritance – Backgrounds – Border Images – Colors – Shadows – Text – Transformations – Transitions – Animations.

WEB ESSENTIALS

1.1 Web Essentials:

Server:

The software that distributes the information and the machine where the information and software reside is called the server.

- provides requested service to client
- e.g., Web server sends requested Web page

Client:

The software that resides on the remote machine, communicates with the server, fetches the information, processes it, and then displays it on the remote machine is called the client.

- initiates contact with server (—speaks first!)
- typically requests service from server
- Web: client implemented in browser

Web server:

Software that delivers Web pages and other documents to browsers using the HTTP protocol

Web Page:

A web page is a document or resource of information that is suitable for the World Wide Web and can be accessed through a web browser.

Website:

A collection of pages on the World Wide Web that are accessible from the same URL and typically residing on the same server.

URL:

Uniform Resource Locator, the unique address which identifies a resource on the Internet for routing purposes.

1.2 Client-server paradigm:

The Client-Server paradigm is the most prevalent model for distributed computing protocols. It is the basis of all distributed computing paradigms at a higher level of abstraction. It is service-oriented, and employs a request-response protocol.

A server process, running on a server host, provides access to a service. A client process, running on a client host, accesses the service via the server process. The interaction of the process proceeds according to a protocol.

The primary idea of a client/server system is that you have a central repository of information—some kind of data, often in a database—that you want to distribute on demand to some set of people or machines.

1.3 The Internet:

- Medium for communication and interaction in inter connected network.
- Makes information constantly and instantly available to anyone with a connection.

Web Browsers:

- User agent for Web is called a browser:
 - o Internet Explorer

- o Firefox

Web Server:

- Server for Web is called Web server:

- o Apache (public domain)

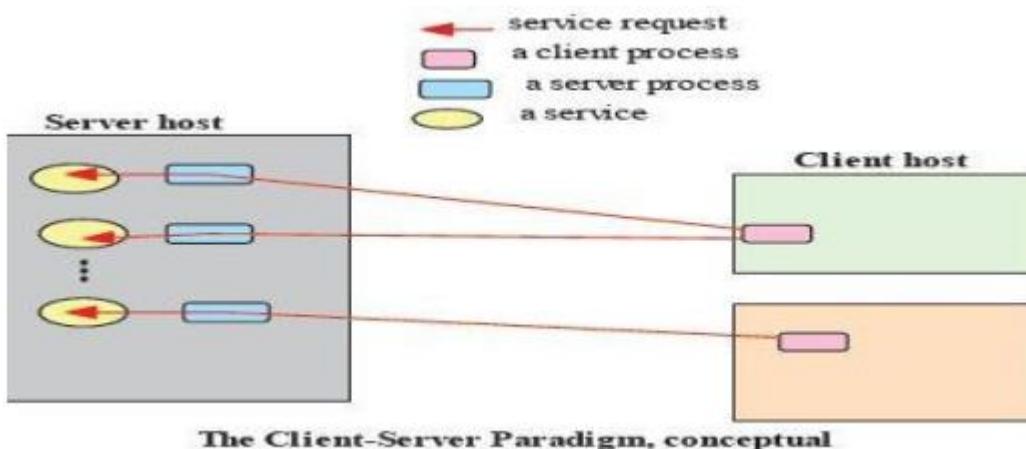
- o MS Internet Information Server

Protocol:

Protocols are agreed formats for transmitting data between devices.

The protocol determines:

- i. The error checking required
- ii. Data compression method used
- iii. The way the end of a message is signalled
- iv. The way the device indicates that it has received the message



1.4 Internet Protocol:

There are many protocols used by the Internet and the WWW:

- o TCP/IP
- o HTTP
- o FTP
- o Electronic mail protocols IMAP
- o POP

TCP/IP

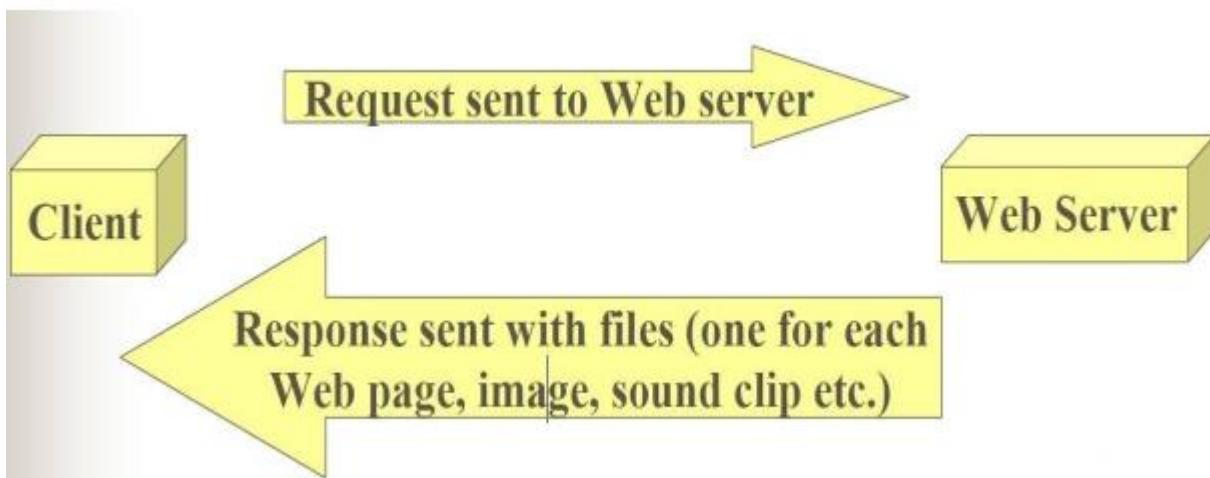
The Internet uses two main protocols (developed by Vincent Cerf and Robert Kahn)

Transmission control protocol (TCP):Controls disassembly of message into packets at the origin reassembles at the destination

Internet protocol (IP):Specifies the addressing details for each packet Each packet is labelled with its origin and destination.

1.5 Hypertext Transfer Protocol (HTTP)

- The hypertext transfer protocol (HTTP) was developed by Tim Berners-Lee in 1991
- HTTP was designed to transfer pages between machines
- The client (or Web browser) makes a request for a given page and the Server is responsible for finding it and returning it to the client
- The browser connects and requests a page from the server
- The server reads the page from the file system, sends it to the client and terminates the connection.



Electronic Mail Protocols:

- Electronic mail uses the client/server model
- The organisation has an email server devoted to handling email
 - Stores and forwards email messages
 - Individuals use email client software to read and send email
 - (e.g. Microsoft Outlook, or Netscape Messenger)
 - Simple Mail Transfer Protocol (SMTP)
 - Specifies format of mail messages
 - Post Office Protocol (POP) tells the email server to:
 - Send mail to the user's computer and delete it from the server
 - Send mail to the user's computer and do not delete it from the server
 - Ask whether new mail has arrived.

1.6 Interactive Mail Access Protocol (IMAP)

Newer than POP, provides similar functions with additional features.
 e.g. can send specific messages to the client rather than all the messages.
 A user can view email message headers and the sender's name before
 downloading the entire message.

Allows users to delete and search mailboxes held on the email server.

The disadvantage of POP: You can only access messages from one PC.

The disadvantage of IMAP : Since email is stored on the email server, there is a need for more and more expensive (high speed) storage space.

1.7 World Wide Web: comprises software (Web server and browser) and data (Web sites).

Internet Protocol (IP) Addresses:

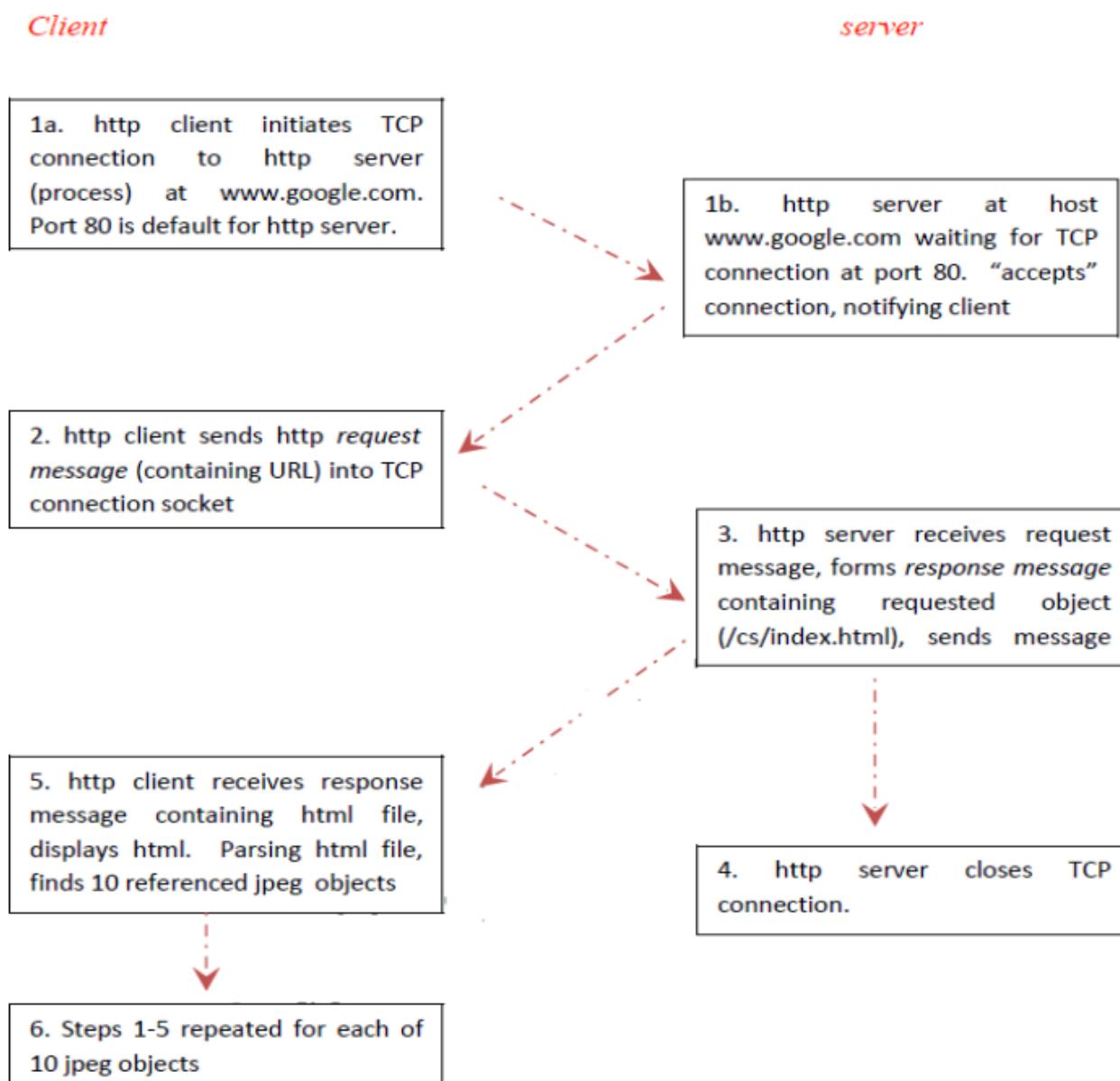
- Every node has a unique numeric address
- Form: 32-bit binary number
- New standard, IPv6, has 128 bits (1998)
- Organizations are assigned groups of IPs for their computers
- **Domain names**
 - Form: host-name. domain-names
 - First domain is the smallest (Google)
 - Last domain specifies the type of organization (.com)
 - Fully qualified domain name - the host name and all of the domain names
 - DNS servers - convert fully qualified domain names to IPs

1.8 HTTP:

- Hypertext Transfer Protocol (HTTP) is the communication protocol used by the Internet to transfer hypertext documents.
- A protocol to transfer hypertext requests and information between servers and browsers

- Hypertext is text, displayed on a computer, with references (hyperlinks) to other text that the reader can immediately follow, usually by a mouse. HTTP is behind every request for a web document or graph, every click of a hypertext link, and every submission of a form.
- HTTP specifies how clients **request** data, and how servers **respond** to these requests.
- The client makes a request for a given page and the server is responsible for finding it and returning it to the client.
- The browser connects and requests a page from the server.
- The server reads the page from the file system and sends it to the client and then terminates the connection

HTTP Transactions

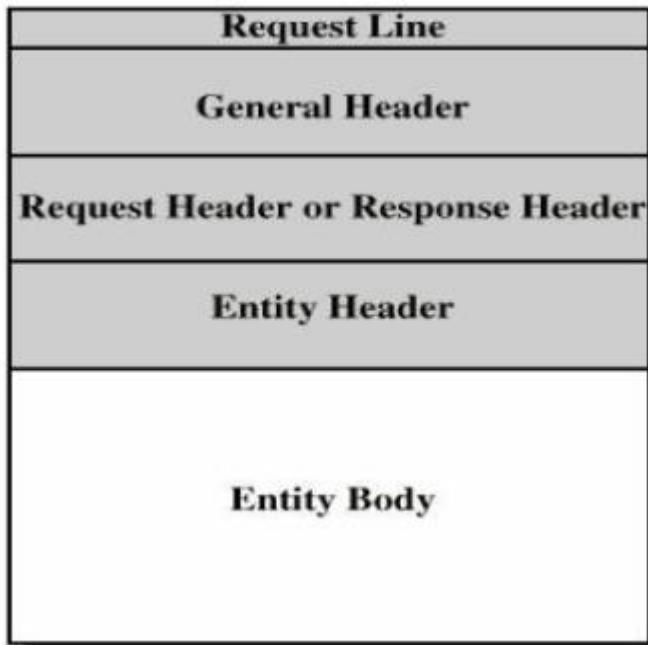


1.9 HTTP Message:

HTTP message is the information transaction between the client and server.

Two types of HTTP Message:

1. Requests
 - a. Client to server
2. Responses
 - a. Server to client



Fields

- Request line or Response line
- General header
- Request header or Response header
- Entity header
- Entity body

.10 Request Message:

Request Line:

- A request line has three parts, separated by spaces
 - o a *method* name
 - o the local path of the requested resource
 - o the version of HTTP being used
- A typical request line is:
 - o GET /path/to/file/index.html HTTP/1.1
- Notes:
 - o **GET** is the most common HTTP method; it says "give me this resource". Other methods include **POST** and **HEAD**. Method names are always uppercase
 - o The path is the part of the URL after the host name, also called the *request URI*
 - o The HTTP version always takes the form "**HTTP/x.x**", uppercase.

Request Header:

HTTP Request Headers

Header	Description
From	Email address of user
User-Agent	Client s/w
Accept File	File types that client will accept
Accept-encoding	Compression methods
Accept-Language	Languages
Referrer	URL of the last document the client displayed
If-Modified-Since	Return document only if modified since specified
Content-length	Length (in bytes) of data to follow

HTTP Request

Method File name HTTP version

```

GET /msaleh/index.html HTTP/1.1
Host: staff.ifm.ac.tz
Connection: close
Accept: text/xml, text/html, text/plain, image/png, */*
Accept-Language: en-GB, en
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Accept-Charset: ISO-8859-1, utf-8; q=0.7, *
If-Modified-Since: Mon, 18 Sep 2006 22:57:19 GMT
Referer: http://web-sniffer.net

```

Blank line

Headers

Data – none for GET

.11 Response Message:

Response Line:

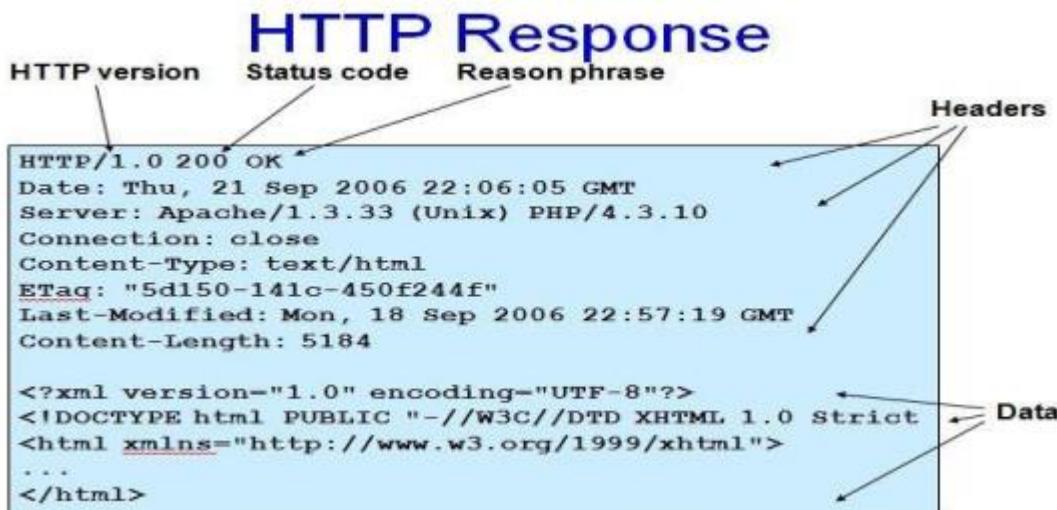
- A request line has three parts, separated by spaces
 - the HTTP version,
 - a *response status code* that gives the result of the request, and
 - an English *reason phrase* describing the status code
- Typical status lines are:
 - HTTP/1.0 200 OK or
 - HTTP/1.0 404 Not Found
 - Notes:
 - The HTTP version is in the same format as in the request line, "HTTP/x.x".

- The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary.

HTTP Request Header:

HTTP Response Headers	
Header	Description
Server	Server software
Date	Current Date
Last-Modified	Modification date of document
Expires	Date at which document expires
Location	The location of the document in redirection responses
Pragma	A hint, e.g., no cache
MIME-version	
Link	URL of document's parent
Content-Length	Length in bytes
Allowed	Requests that user can issue, e.g., GET

EXAMPLE



HTTP Method:

- HTTP method is supplied in the request line and specifies the operation that the client has requested.

Some common methods:

- Options
- Get
- Head
- Post
- Put
- Move
- Delete

Two methods that are mostly used are the GET and POST:

- GET for queries that can be safely repeated

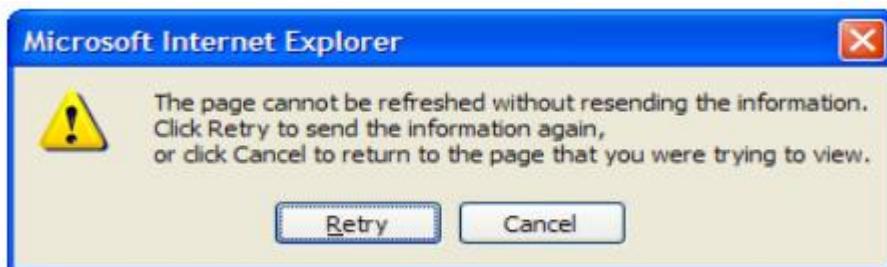
- o **POST** for operations that may have side effects (e.g. ordering a book from an on-line store).

The GET Method

- It is used to retrieve information from a specified URI and is assumed to be a safe, repeatable operation by browsers, caches and other HTTP aware components
- Operations have no side effects and GET requests can be re-issued.
- For example, displaying the balance of a bank account has no effect on the account and can be safely repeated.
- Most browsers will allow a user to refresh a page that resulted from a **GET**, without displaying any kind of warning
- Proxies may automatically retry **GET** requests if they encounter a temporary network connection problem.
- GET requests is that they can only supply data in the form of parameters encoded in the URI (known as a **Query String**) – [downside]
- Cannot be unused for uploading files or other operations that require large amounts of data to be sent to the server.

The POST Method

- Used for operations that have side effects and cannot be safely repeated.
- For example, transferring money from one bank account to another has side effects and should not be repeated without explicit approval by the user.
- If you try to refresh a page in Internet Explorer that resulted from a **POST**, it displays the following message to warn you that there may



The **POST** request message has a content body that is normally used to send parameters and data

- The IIS server returns two status codes in its response for a **POST** request
 - o The first is **100 Continue** to indicate that it has successfully received the **POST** request
 - o The second is **200 OK** after the request has been processed.

HTTP response status codes

- Informational (1xx)
- Successful (2xx)
- Redirection (3xx)
 - o 301: moved permanently
 - Client error (4xx)
 - o 403 : forbidden
 - o 404: Not found
 - Server error (5xx)
 - o 503: Service unavailable
 - o 505: HTTP version not supported

1.12 HTTP

❖ □ Features

- Persistent TCP Connections: Remain open for multiple requests
- Partial Document Transfers: Clients can specify start and stop positions
- Conditional Fetch: Several additional conditions

- Better content negotiation
- More flexible authentication.

❖ □ Web Browsers :

- Mosaic - NCSA (Univ. of Illinois), in early 1993 First to use a GUI, led to Explosion of Web use Initially for X-Windows, under UNIX, but was ported to other platforms by late 1993
- Browsers are clients - always initiate, servers react (although sometimes servers require responses)
- Most requests are for existing documents, using Hypertext Transfer Protocol (HTTP)
- But some requests are for program execution, with the output being returned as a document.

Browser: A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.

❖ □ Web Servers:

- Provide responses to browser requests, either existing documents or dynamically built documents.
- Browser-server connection is now maintained through more than one request- Response cycle
- All communications between browsers and servers use Hypertext Transfer Protocol
- Web servers run as background processes in the operating system.
- Monitor a communications port on the host, accepting HTTP messages when they appear

All current Web servers came from either

1. The original from CERN
 2. The second one, from NCSA
- Web servers have two main directories:
 1. Document root (servable documents)
 2. Server root (server system software)
 - Document root is accessed indirectly by clients
 - Its actual location is set by the server Configuration file
 - Requests are mapped to the actual location
 - Virtual document trees
 - Virtual hosts
 - Proxy servers
 - Web servers now support other Internet protocols
 - Apache (open source, fast, reliable)
 - IIS
 - Maintained through a program with a GUI interface.

HTML 5

HTML is the main markup language for describing the structure of web pages.

HTML stands for HyperText Markup Language. HTML is the basic building block of World Wide Web.

Hypertext is text displayed on a computer or other electronic device with references to other text that the user can immediately access, usually by a mouse click or key press.

Apart from text, hypertext may contain tables, lists, forms, images, and other presentational elements. It is an easy-to-use and flexible format to share information over the Internet.

Markup languages use sets of markup tags to characterize text elements within a document, which gives instructions to the web browsers on how the document should appear.

HTML was originally developed by Tim Berners-Lee in 1990. He is also known as the father of the web. In 1996, the World Wide Web Consortium (W3C) became the authority to maintain the HTML specifications. HTML also became an international standard (ISO) in 2000. HTML5 is the latest version of HTML. HTML5 provides a faster and more robust approach to web development.

HTML Tags and Elements

HTML is written in the form of HTML elements consisting of markup tags. These markup tags are the fundamental characteristic of HTML. Every markup tag is composed of a keyword, surrounded by angle brackets, such as `<html>`, `<head>`, `<body>`, `<title>`, `<p>`, and so on.

HTML tags normally come in pairs like `<html>` and `</html>`. The first tag in a pair is often called the opening tag (or start tag), and the second tag is called the closing tag (or end tag).

An opening tag and a closing tag are identical, except a slash (/) after the opening angle bracket of the closing tag, to tell the browser that the command has been completed.

Inserting Images into Web Pages

Images enhance visual appearance of the web pages by making them more interesting and colorful.

The `` tag is used to insert images in the HTML documents. It is an empty element and contains attributes only. The syntax of the `` tag can be given with:

```

```

The following example inserts three images on the web page:

Example

[Try this code »](#)

```



```

Each image must carry at least two attributes: the `src` attribute, and an `alt` attribute.

The `src` attribute tells the browser where to find the image. Its value is the URL of the image file.

Whereas, the `alt` attribute provides an alternative text for the image, if it is unavailable or cannot be displayed for some reason. Its value should be a meaningful substitute for the image.

HTML Tables

Creating Tables in HTML

HTML table allows you to arrange data into rows and columns. They are commonly used to display tabular data like product listings, customer's details, financial reports, and so on.

You can create a table using the `<table>` element. Inside the `<table>` element, you can use the `<tr>` elements to create rows, and to create columns inside a row you can use the `<td>` elements. You can also define a cell as a header for a group of table cells using the `<th>` element.

The following example demonstrates the most basic structure of a table.

Example

```
<table>
  <tr>
    <th>No.</th>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Peter Parker</td>
    <td>16</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Clark Kent</td>
    <td>34</td>
  </tr>
</table>
```

Tables do not have any borders by default. You can use the CSS `border` property to add borders to the tables. Also, table cells are sized just large enough to fit the contents by default. To add more space around the content in the table cells you can use the CSS `padding` property.

Defining a Table Header, Body, and Footer

HTML provides a series of tags `<thead>`, `<tbody>`, and `<tfoot>` that helps you to create more structured table, by defining header, body and footer regions, respectively.

The following example demonstrates the use of these elements.

Example

```
<table>
  <thead>
    <tr>
      <th>Items</th>
      <th>Expenditure</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Stationary</td>
      <td>2,000</td>
    </tr>
    <tr>
      <td>Furniture</td>
      <td>10,000</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <th>Total</th>
      <td>12,000</td>
    </tr>
  </tfoot>
```

```
</tr>
</tfoot>
</table>
```

HTML Lists

HTML lists are used to present list of information in well formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning.

- **Unordered list** — Used to create a list of related items, in no particular order.
- **Ordered list** — Used to create a list of related items, in a specific order.
- **Description list** — Used to create a list of terms and their descriptions.

HTML Unordered Lists

An unordered list created using the `` element, and each list item starts with the `` element.

The list items in unordered lists are marked with bullets. Here's an example:

Example

```
<ul>
  <li>Chocolate Cake</li>
  <li>Black Forest Cake</li>
  <li>Pineapple Cake</li>
</ul>
```

— The output of the above example will look something like this:

- Chocolate Cake
- Black Forest Cake
- Pineapple Cake

You can also change the bullet type in your unordered list using the CSS `list-style-type` property. The following style rule changes the type of bullet from the default `disc` to `square`:

Example

```
ul {
  list-style-type: square;
}
```

Please check out the tutorial on [CSS lists](#) to learn about styling HTML lists in details.

HTML Ordered Lists

An ordered list created using the `` element, and each list item starts with the `` element. Ordered lists are used when the order of the list's items is important.

The list items in an ordered list are marked with numbers. Here's an example:

Example

```
<ol>
  <li>Fasten your seatbelt</li>
  <li>Starts the car's engine</li>
  <li>Look around and go</li>
</ol>
```

— The output of the above example will look something like this:

1. Fasten your seatbelt
2. Starts the car's engine
3. Look around and go

HTML5 Image

HTML Images Syntax

In HTML, images are defined with the `` tag.

The `` tag is empty, it contains attributes only, and does not have a closing tag.

The `src` attribute specifies the URL (web address) of the image:

```

```

EXAMPLE

```
<!DOCTYPE html>

<html>

<body>

<h2>HTML Image</h2>



</body>

</body>

</html>
```

OUTPUT

HTML Image



HTML Form

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called **controls** like *inputbox*, *checkboxes*, *radio-buttons*, *submit buttons*, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

The `<form>` tag is used to create an HTML form. Here's a simple example of a login form:

Example

```
<form>
  <label>Username: <input type="text"></label>
  <label>Password: <input type="password"></label>
  <input type="submit" value="Submit">
</form>
```

The following section describes different types of controls that you can use in your form.

Input Element

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the `type` attribute. An input element can be of type *text field*, *password field*, *checkbox*, *radio button*, *submit button*, *reset button*, *file select box*, as well as several [new input types](#) introduced in HTML5.

The most frequently used input types are described below.

Text Fields

Text fields are one line areas that allow the user to input text.

Single-line text input controls are created using an `<input>` element, whose `type` attribute has a value of `text`. Here's an example of a single-line text input used to take username:

Example

```
<form>
  <label for="username">Username:</label>
  <input type="text" name="username" id="username">
</form>
```

— The output of the above example will look something like this:

Username:

Password Field

Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots. This is to prevent someone else from reading the password on the screen. This is also a single-line text input controls created using an `<input>` element whose `type` attribute has a value of `password`.

Example

```
<form>
  <label for="user-pwd">Password:</label>
  <input type="password" name="user-password" id="user-pwd">
</form>
```

— The output of the above example will look something like this:

Password:

Radio Buttons

Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `radio`.

Example

[Try this code »](#)

```
<form>
  <input type="radio" name="gender" id="male">
  <label for="male">Male</label>
  <input type="radio" name="gender" id="female">
  <label for="female">Female</label>
</form>
```

— The output of the above example will look something like this:

Male Female

Checkboxes

Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `checkbox`.

Example

```
<form>
  <input type="checkbox" name="sports" id="soccer">
  <label for="soccer">Soccer</label>
  <input type="checkbox" name="sports" id="cricket">
```

```
<label for="cricket">Cricket</label>
<input type="checkbox" name="sports" id="baseball">
<label for="baseball">Baseball</label>
</form>
```

— The output of the above example will look something like this:

Soccer Cricket Baseball

File Select box

The file fields allow a user to browse for a local file and send it as an attachment with the form data. Web browsers such as Google Chrome and Firefox render a file select input field with a Browse button that enables the user to navigate the local hard drive and select a file.

This is also created using an `<input>` element, whose `type` attribute value is set to `file`.

Example

```
<form>
  <label for="file-select">Upload:</label>
  <input type="file" name="upload" id="file-select">
</form>
```

— The output of the above example will look something like this:

Upload: No file chosen

Textarea

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an `<textarea>` element.

Example

```
<form>
  <label for="address">Address:</label>
  <textarea rows="3" cols="30" name="address" id="address"></textarea>
</form>
```

— The output of the above example will look something like this:

Address: 

Select Boxes

A select box is a dropdown list of options that allows user to select one or more option from a pull-down list of options. Select box is created using the `<select>` element and `<option>` element.

The `<option>` elements within the `<select>` element define each list item.

Example

```
<form>
  <label for="city">City:</label>
  <select name="city" id="city">
    <option value="sydney">Sydney</option>
    <option value="melbourne">Melbourne</option>
    <option value="cromwell">Cromwell</option>
  </select>
</form>
```

— The output of the above example will look something like this:

City:

Submit and Reset Buttons

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's `action` attribute to process the submitted data.

A reset button resets all the forms control to default values. Try out the following example by typing your name in the text field, and click on submit button to see it in action.

Example

```
<form action="action.php" method="post">
  <label for="first-name">First Name:</label>
  <input type="text" name="first-name" id="first-name">
  <input type="submit" value="Submit">
  <input type="reset" value="Reset">
</form>
```

First Name:

HTML5 Colors

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

<h1 style="background-color:Tomato;">Tomato</h1>
<h1 style="background-color:Orange;">Orange</h1>
<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>
<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>
```

```
<h1 style="background-color:Gray;">Gray</h1>  
<h1 style="background-color:SlateBlue;">SlateBlue</h1>  
<h1 style="background-color:Violet;">Violet</h1>  
<h1 style="background-color:LightGray;">LightGray</h1>  
  
</body>  
</html>
```

OUTPUT



The image shows a vertical stack of colored bars, each representing the background color of an H1 tag from the provided code. From top to bottom, the colors are: Tomato (red), Orange (yellow), DodgerBlue (blue), MediumSeaGreen (green), Gray (gray), SlateBlue (purple), Violet (pink), and LightGray (light gray). The bars are separated by thin white lines.

Tomato
Orange
DodgerBlue
MediumSeaGreen
Gray
SlateBlue
Violet
LightGray

HTML5 Audio

Embedding Audio in HTML Document

Inserting audio onto a web page was not easy before, because web browsers did not have a uniform standard for defining embedded media files like audio.

Using the HTML5 audio Element

The newly introduced HTML5 `<audio>` element provides a standard way to embed audio in web pages. However, the audio element is relatively new but it works in most of the modern web browsers.

The following example simply inserts an audio into the HTML5 document, using the browser default set of controls, with one source defined by the `src` attribute.

Example

```
<audio controls="controls" src="media/birds.mp3">
  Your browser does not support the HTML5 Audio element.
</audio>
```

An audio, using the browser default set of controls, with alternative sources.

Example

```
<audio controls="controls">
  <source src="media/birds.mp3" type="audio/mpeg">
  <source src="media/birds.ogg" type="audio/ogg">
  Your browser does not support the HTML5 Audio element.
</audio>
```

HTML5 Video

Embedding Video in HTML Document

Inserting video onto a web page was not relatively easy, because web browsers did not have a uniform standard for defining embedded media files like video.

Using the HTML5 video Element

The newly introduced HTML5 `<video>` element provides a standard way to embed video in web pages. However, the video element is relatively new, but it works in most of the modern web browsers.

The following example simply inserts a video into the HTML document, using the browser default set of controls, with one source defined by the `src` attribute.

Example

```
<video controls="controls" src="media/shuttle.mp4">
  Your browser does not support the HTML5 Video element.
</video>
```

A video, using the browser default set of controls, with alternative sources.

Example

```
<video controls="controls">
  <source src="media/shuttle.mp4" type="video/mp4">
  <source src="media/shuttle.ogv" type="video/ogg">
  Your browser does not support the HTML5 Video element.
</video>
```

New HTML5 Elements

The most interesting new HTML5 elements are:

New **semantic elements** like `<header>`, `<footer>`, `<article>`, and `<section>`.

New **attributes of form elements** like number, date, time, calendar, and range.

New **graphic elements**: `<svg>` and `<canvas>`.

New **multimedia elements**: `<audio>` and `<video>`.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: `<div>` and `` - Tells nothing about its content.

Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

New Semantic Elements in HTML5

Many web sites contain HTML code like:

```
<div id="nav"><div class="header"><div id="footer">  
to indicate navigation, header, and footer.
```

HTML5 offers new semantic elements to define different parts of a web page:

- `<article>`
- `<aside>`
- `<details>`
- `<figcaption>`
- `<figure>`
- `<footer>`
- `<header>`
- `<main>`
- `<mark>`
- `<nav>`
- `<section>`
- `<summary>`
- `<time>`



HTML5 <section> Element

The `<section>` element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

A home page could normally be split into sections for introduction, content, and contact information.

Example

```

<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>

```

HTML5 <article> Element

The `<article>` element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an `<article>` element can be used:

- Forum post
- Blog post
- Newspaper article

Example

```

<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>

```

HTML5 <header> Element

The `<header>` element specifies a header for a document or section.

The `<header>` element should be used as a container for introductory content.

You can have several `<header>` elements in one document.

The following example defines a header for an article:

Example

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML5 `<footer>` Element

The `<footer>` element specifies a footer for a document or section.

A `<footer>` element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You may have several `<footer>` elements in one document.

Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

HTML5 `<figure>` and `<figcaption>` Elements

The purpose of a figure caption is to add a visual explanation to an image.

In HTML5, an image and a caption can be grouped together in a `<figure>` element:

Example

```
<figure>
  
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

OUTPUT

Places to Visit

Puglia's most famous sight is the unique conical houses (Trulli) found in the area around Alberobello, a declared UNESCO World Heritage Site.



Fig.1 - Trulli, Puglia, Italy.

Semantic Elements in HTML5

Below is an alphabetical list of the new semantic elements in HTML5.

The links go to our complete [HTML5 Reference](#).

Tag	Description
<code><article></code>	Defines an article
<code><aside></code>	Defines content aside from the page content
<code><details></code>	Defines additional details that the user can view or hide
<code><figcaption></code>	Defines a caption for a <code><figure></code> element

<u><figure></u>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<u><footer></u>	Defines a footer for a document or section
<u><header></u>	Specifies a header for a document or section
<u><main></u>	Specifies the main content of a document
<u><mark></u>	Defines marked/highlighted text
<u><nav></u>	Defines navigation links
<u><section></u>	Defines a section in a document
<u><summary></u>	Defines a visible heading for a <details> element
<u><time></u>	Defines a date/time

HTML5 Drag and Drop



Drag the W3Schools image into the rectangle.

Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

In HTML5, drag and drop is part of the standard: Any element can be draggable.

HTML Drag and Drop Example

The example below is a simple drag and drop example:

Example

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>



</body>
</html>
```

OUTPUT

Drag the W3Schools image into the rectangle:



HTML5 <nav> Element

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a `<nav>` element. The `<nav>` element is intended only for major block of navigation links.

Example

```
<nav>
<a href="/html/">HTML</a> |
<a href="/css/">CSS</a> |
<a href="/js/">JavaScript</a> |
<a href="/jquery/">jQuery</a>
</nav>
```

What Is CSS3 And Why Is It Used?

To help build highly interactive online pages, CSS3 is invariably used due to its importance in providing greater options in the design process. When marketing products and services, web design plays a vital part; a site should be created in a manner that will draw potential customers to explore and revisit a website more often. Many **web design firms** are developing and enhancing websites through the use of CSS3 as this is a great form of web development. This article will help define CSS3 and will point out its advantages.

Definition

The acronym CSS stands for Cascading Style Sheets which is used to augment the functionality, versatility, and efficient performance of site content. It allows for the creation of content-rich websites that do not require much weight or codes; this translates into more interactive graphics and animation, superior user-interface, and significantly more organization and rapid download time.

It is used with HTML to create content structure, with CSS3 being used to format structured content. It is responsible for font properties, colors, text alignments, graphics, background images, tables and other components. This tool provides extra capabilities such as absolute, fixed and relative positioning of various elements. The increasing popularity of CSS3 when used by **web design firms** stimulates major browsers such as Google Chrome, Firefox, Safari, and IE9 to adopt and embrace this programming language.

Advantages

Although CSS3 is not the only web development solution, it does allow provide greater advantages for several reasons.

- **Customization** – A web page can be customized and alterations created in the design by simply changing a modular file.
- **Bandwidth Requirements** – It decreases server bandwidth requirements, giving rapid download time when a site is accessed with desktop or hand-held devices, providing an improved user experience.
- **Consistency** – It delivers consistent and accurate positioning of navigational elements on the website.
- **Appealing** – It makes the site more appealing with adding videos and graphics easier.
- **Viewing** – It allows online videos to be viewed without the use of third-party plug-ins.
- **Visibility** – It delivers the opportunity to improve brand visibility by designing effective online pages.
- **Cost Effective** – It is cost-effective, time-saving, and supported by most browsers.

Since the introduction of CSS3, there is greater control of the presentation of content and various elements on a website; however it is not really responsible for overall design as it only specifies the structure and content presentation of certain web pages.

External, internal, and inline CSS styles

Cascading Style Sheets (CSS) are files with styling rules that govern how your website is presented on screen. CSS rules can be applied to your website's HTML files in various ways. You can use an **external stylesheet**, an **internal stylesheet**, or an **inline style**. Each method has advantages that suit particular uses.

An **external stylesheet** is a standalone .css file that is linked from a web page. The advantage of external stylesheets is that it can be created once and the rules applied to multiple web pages. Should you need to make widespread changes to your site design, you can make a single change in the stylesheet and it will be applied to all linked pages, saving time and effort.

An **internal stylesheet** holds CSS rules for the page in the **head** section of the HTML file. The rules only apply to that page, but you can configure CSS classes and IDs that can be used to style multiple elements in the page code. Again, a single change to the CSS rule will apply to all tagged elements on the page.

Inline styles relate to a specific HTML tag, using a **style** attribute with a CSS rule to style a specific page element. They're useful for quick, permanent changes, but are less flexible than external and internal stylesheets as each inline style you create must be separately edited should you decide to make a design change.

Using external CSS stylesheets

An HTML page styled by an external CSS stylesheet must reference the .css file in the document head. Once created, the CSS file must be uploaded to your server and linked in the HTML file with code such as:

```
<link href="style.css" rel="stylesheet" type="text/css">
```

You can name your stylesheet whatever you wish, but it should have a .css file extension.

Using internal CSS stylesheets

Rather than linking an external .css file, HTML files using an internal stylesheet include a set of rules in their **head** section. CSS rules are wrapped in **<style>** tags, like this:

```
<head>
<style type="text/css">

h1 {
    color:#fff
    margin-left: 20px;
}

p {
```

```
font-family: Arial, Helvetica, Sans Serif;  
}  
  
</style>  
</head>
```

Using inline styles

Inline styles are applied directly to an element in your HTML code. They use the **style** attribute, followed by regular CSS properties.

For example:

```
<h1 style="color:red;margin-left:20px;">Today's Update</h1>
```

Rule Cascading

Cascade and inheritance

Conflicting rules

CSS stands for **Cascading Style Sheets**, and that first word *cascading* is incredibly important to understand — the way that the cascade behaves is key to understanding CSS.

At some point, we will find that the CSS have created two rules which could potentially apply to the same element. The **cascade**, and the closely-related concept of **specificity**, are mechanisms that control which rule applies when there is such a conflict. Which rule is styling your element may not be the one you expect, so you need to understand how these mechanisms work.

Also significant here is the concept of **inheritance**, which means that some CSS properties by default inherit values set on the current element's parent element, and some don't. This can also cause some behavior that you might not expect.

The cascade

Stylesheets **cascade** — at a very simple level this means that the order of CSS rules matter; when two rules apply that have equal specificity the one that comes last in the CSS is the one that will be used.

EXAMPLE

In the below example, we have two rules that could apply to the h1. The h1 ends up being colored blue — these rules have an identical selector and therefore carry the same specificity, so the last one in the source order wins.

```
h1 {  
  color: red;  
}
```

```
}
```

```
h1 {
```

```
    color: blue;
```

```
}
```

```
<h1>This is my heading.</h1>
```

OUTPUT

```
This is my heading.
```

Specificity

Specificity is how the browser decides which rule applies if multiple rules have different selectors, but could still apply to the same element. It is basically a measure of how specific a selector's selection will be:

- An element selector is less specific — it will select all elements of that type that appear on a page — so will get a lower score.
- A class selector is more specific — it will select only the elements on a page that have a specific class attribute value — so will get a higher score.

Example time! Below we again have two rules that could apply to the h1. The below h1 ends up being colored red — the class selector gives its rule a higher specificity, and so it will be applied even though the rule with the element selector appears further down in the source order.

EXAMPLE

```
main-heading {
```

```
    color: red;
```

```
}
```

```
h1 {
```

```
    color: blue;
```

```
}
```

```
. <h1 class="main-heading">This is my heading.</h1>
```

OUTPUT

```
This is my heading.
```

Inheritance

Inheritance also needs to be understood in this context — some CSS property values set on parent elements are inherited by their child elements, and some aren't.

For example, if you set a color and font-family on an element, every element inside it will also be styled with that color and font, unless you've applied different color and font values directly to them.

Some properties do not inherit — for example if you set a `width` of 50% on an element, all of its descendants do not get a width of 50% of their parent's width. If this was the case, CSS would be very frustrating to use!

```
body {  
  color: blue;  
}  
  
span {  
  color: black;  
}
```

```
<p>As the body has been set to have a color of blue this is inherited through the  
descendants.</p>
```

```
<p>We can change the color by targetting the element with a selector,  
such as this
```

```
<span>span</span>.</p>
```

OUTPUT

```
As the body has been set to have a color of blue this is inherited through  
the descendants.
```

```
We can change the color by targetting the element with a selector, such as  
this span.
```

CSS3 Shadow Effects

With CSS3 you can create two types of shadows: `text-shadow` (adds shadow to text) and `box-shadow` (adds shadow to other elements).

CSS3 Text Shadow

The `text-shadow` property can take up to four values:

- the horizontal shadow
- the vertical shadow
- the blur effect
- the color

Examples:

- Normal text shadow

```
• h1 {  
  • text-shadow: 2px 2px 5px crimson;
```

```
}
```

CSS3 Text Shadow Effect

- Glowing text effect

```
• h1 {  
  • text-shadow: 0 0 4px #00FF9C;
```

```
}
```

This Title Glows!

CSS3 Box Shadow

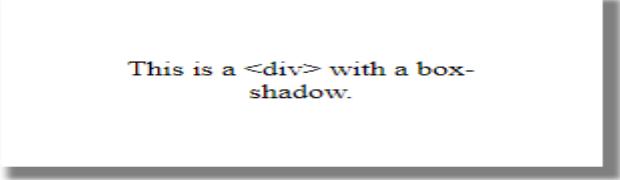
The `box-shadow` property can take up to six values:

- (optional) the `inset` keyword (changes the shadow to one inside the frame)
- the horizontal shadow
- the vertical shadow
- the blur effect
- the spreading
- the color

Examples:

```
.first-div {
    box-shadow: 1px 1px 5px 3px grey;
}
```

This is a <div> with a box-shadow.



CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

What are CSS Animations?

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

When you specify CSS styles inside the `@keyframes` rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

Example

```
/* The animation code */
@keyframes example {
```

```

from {background-color: red;}
to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}

```

Note: The `animation-duration` property defines how long time an animation should take to complete. If the `animation-duration` property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

Example

```

/* The animation code */
@keyframes example {
  0% {background-color: red;}
  25% {background-color: yellow;}
  50% {background-color: blue;}
  100% {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}

```

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	Description
@keyframes	Specifies the animation code
animation	A shorthand property for setting all the animation properties

<u>animation-delay</u>	Specifies a delay for the start of an animation
<u>animation-direction</u>	Specifies whether an animation should be played forwards, backwards or in alternate cycles
<u>animation-duration</u>	Specifies how long time an animation should take to complete one cycle
<u>animation-fill-mode</u>	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
<u>animation-iteration-count</u>	Specifies the number of times an animation should be played
<u>animation-name</u>	Specifies the name of the @keyframes animation
<u>animation-play-state</u>	Specifies whether the animation is running or paused
<u>animation-timing-function</u>	Specifies the speed curve of the animation

CSS Transitions

CSS Transitions is a module of CSS that lets you create gradual transitions between the values of specific CSS properties. The behavior of these transitions can be controlled by specifying their timing function, duration, and other attributes.

Properties

- [transition](#)
- [transition-delay](#)
- [transition-duration](#)
- [transition-property](#)
- [transition-timing-function](#)

The **transition** [CSS](#) property is a [shorthand property](#) for [transition-property](#), [transition-duration](#), [transition-timing-function](#), and [transition-delay](#).

CSS transition Property

Example

Hover over a <div> element to gradually change the width from 100px to 300px:

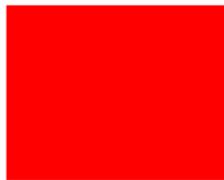
```
div {
  width: 100px;
  transition: width 2s;
}

div:hover {
  width: 300px;
}
```

OUTPUT

The transition Property

Hover over the div element below, to see the transition effect:



Definition and Usage

The **transition** property is a shorthand property for:

- [transition-property](#)
- [transition-duration](#)
- [transition-timing-function](#)

Property Values

Value	Description
transition-property	Specifies the name of the CSS property the transition effect is for
transition-duration	Specifies how many seconds or milliseconds the transition effect takes to complete
transition-timing-function	Specifies the speed curve of the transition effect
transition-delay	Defines when the transition effect will start
initial	Sets this property to its default value. Read about initial
inherit	Inherits this property from its parent element. Read about inherit

Example

When an <input type="text"> gets focus, gradually change the width from 100px to 250px:

```
input[type=text] {  
    width: 100px;  
    transition: width .35s ease-in-out;  
}  
  
input[type=text]:focus {
```

```
width: 250px;  
}
```

OUTPUT

The width Property

Set the width of the input field to 100 pixels. However, when the input field gets focus, make it 250 pixels wide:

Search:

CSS background-color

The `background-color` property specifies the background color of an element.

Example

The background color of a page is set like this:

```
body {  
    background-color: lightblue;  
}
```

CSS background-image

The `background-image` property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

Example

The background image for a page can be set like this:

```
body {  
    background-image: url("paper.gif");  
}
```

CSS background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

The shorthand property for background is `background`.

Example

Use the shorthand property to set all the background properties in one declaration:

```
body {  
    background: #ffffff url("img_tree.png") no-repeat right top;  
}
```

CSS Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The `border` property is a shorthand property for the following individual border properties:

- `border-width`
- `border-style` (required)
- `border-color`

Example

```
p {  
    border: 5px solid red;  
}
```

Result:

Some text

UNIT II - CLIENT SIDE PROGRAMMING

Java Script: An introduction to JavaScript – JavaScript DOM Model – Date and Objects – Regular Expressions – Exception Handling – Validation – Built-in objects – Event Handling – DHTML with JavaScript – JSON introduction – Syntax – Function Files – Http Request – SQL.

PART - A

Q.No	Questions																						
1.	Evaluate various Java Script Object models.																						
2.	Define DOM.																						
3.	<p>Give any four methods of Date objects.</p> <p><u>JavaScript Get Date Methods</u></p> <p>These methods can be used for getting information from a date object:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 5px;">Method</th><th style="text-align: left; padding: 5px;">Description</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;">getFullYear()</td><td style="padding: 5px;">Get the year as a four digit number (yyyy)</td></tr> <tr> <td style="padding: 5px;">getMonth()</td><td style="padding: 5px;">Get the month as a number (0-11)</td></tr> <tr> <td style="padding: 5px;">getDate()</td><td style="padding: 5px;">Get the day as a number (1-31)</td></tr> <tr> <td style="padding: 5px;">getHours()</td><td style="padding: 5px;">Get the hour (0-23)</td></tr> <tr> <td style="padding: 5px;">getMinutes()</td><td style="padding: 5px;">Get the minute (0-59)</td></tr> <tr> <td style="padding: 5px;">getSeconds()</td><td style="padding: 5px;">Get the second (0-59)</td></tr> <tr> <td style="padding: 5px;">getMilliseconds()</td><td style="padding: 5px;">Get the millisecond (0-999)</td></tr> <tr> <td style="padding: 5px;">getTime()</td><td style="padding: 5px;">Get the time (milliseconds since January 1, 1970)</td></tr> <tr> <td style="padding: 5px;">getDay()</td><td style="padding: 5px;">Get the weekday as a number (0-6)</td></tr> <tr> <td style="padding: 5px;">Date.now()</td><td style="padding: 5px;">Get the time. ECMAScript 5.</td></tr> </tbody> </table> <p><u>JavaScript Set Date Methods</u></p> <p>Set Date methods let you set date values (years, months, days, hours, minutes, seconds, milliseconds) for a Date Object.</p>	Method	Description	getFullYear()	Get the year as a four digit number (yyyy)	getMonth()	Get the month as a number (0-11)	getDate()	Get the day as a number (1-31)	getHours()	Get the hour (0-23)	getMinutes()	Get the minute (0-59)	getSeconds()	Get the second (0-59)	getMilliseconds()	Get the millisecond (0-999)	getTime()	Get the time (milliseconds since January 1, 1970)	getDay()	Get the weekday as a number (0-6)	Date.now()	Get the time. ECMAScript 5.
Method	Description																						
getFullYear()	Get the year as a four digit number (yyyy)																						
getMonth()	Get the month as a number (0-11)																						
getDate()	Get the day as a number (1-31)																						
getHours()	Get the hour (0-23)																						
getMinutes()	Get the minute (0-59)																						
getSeconds()	Get the second (0-59)																						
getMilliseconds()	Get the millisecond (0-999)																						
getTime()	Get the time (milliseconds since January 1, 1970)																						
getDay()	Get the weekday as a number (0-6)																						
Date.now()	Get the time. ECMAScript 5.																						

Set Date Methods

Set Date methods are used for setting a part of a date:

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

	<p>Write the JavaScript methods to retrieve the data and time based on the computer locale.</p> <p><script></p> <pre>// Use of Date.now() function var d = Date(Date.now());</pre> <p>4. // Converting the number of millisecond in date string a = d.toString()</p> <pre>// Printing the current date document.write("The current date is: " + a)</pre> <p></script></p> <p>OUTPUT</p> <p>The current date is: Thu Jan 09 2020 20:35:39 GMT+0530 (India Standard Time)</p>
5.	Can you list the different methods defined in document and window object of JavaScript.

Window Object

The window object represents an open window in a browser.

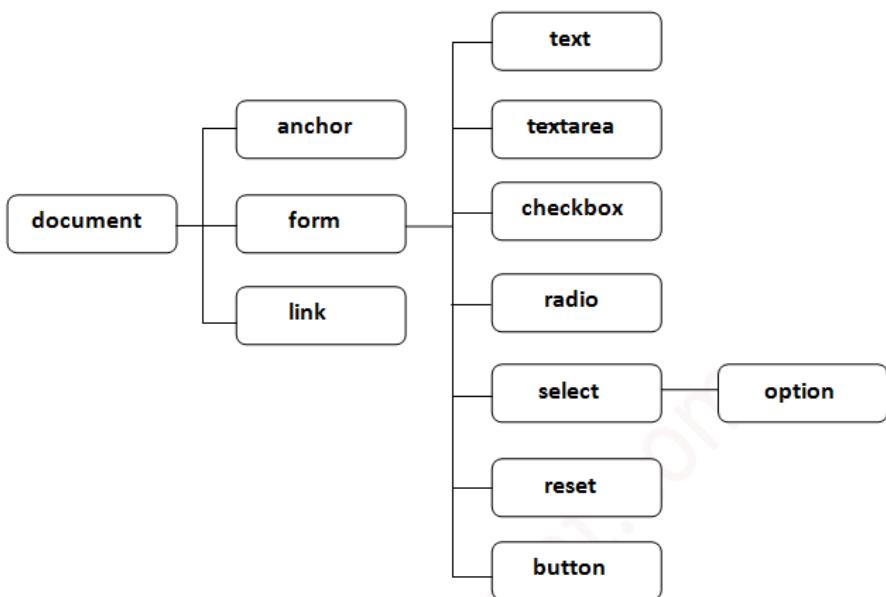
Window Object Methods

Method	Description
alert()	Displays an alert box with a message and an OK button
atob()	Decodes a base-64 encoded string
blur()	Removes focus from the current window
btoa()	Encodes a string in base-64
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
focus()	Sets focus to the current window
getComputedStyle()	Gets the current computed CSS styles applied to an element
getSelection()	Returns a Selection object representing the range of text selected by the user
matchMedia()	Returns a MediaQueryList object representing the specified CSS media query string
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input

	<code>requestAnimationFrame()</code>	Requests the browser to call a function to update an animation before the next repaint
	<code>resizeBy()</code>	Resizes the window by the specified pixels
	<code>resizeTo()</code>	Resizes the window to the specified width and height
	<code>scroll()</code>	Deprecated. This method has been replaced by the <code>scrollTo()</code> method.
	<code>scrollBy()</code>	Scrolls the document by the specified number of pixels
	<code>scrollTo()</code>	Scrolls the document to the specified coordinates
	<code>setInterval()</code>	Calls a function or evaluates an expression at specified intervals (in milliseconds)
	<code>setTimeout()</code>	Calls a function or evaluates an expression after a specified number of milliseconds
	<code>stop()</code>	Stops the window from loading

Document Object Model

1. [Document Object](#)
 2. [Properties of document object](#)
 3. [Methods of document object](#)
 4. [Example of document object](#)
- The **document object** represents the whole html document.
 - When html document is loaded in the browser, it becomes a document object.
 - It is the **root element** that represents the html document. It has properties and methods.
 - By the help of document object, we can add dynamic content to our web page
 - According to W3C - "*The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.*"



Properties of document object

Let's see the properties of document object that can be accessed and modified by the document object.

Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the document.
writeln("string")	writes the given string on the document with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

6.	<p>Name which parser is best in parsing in large size documents. Why?</p>
7.	<p>Summarize benefits of using JavaScript code in an HTML document.</p> <h2>Advantages of JavaScript</h2> <p>The merits of using JavaScript are –</p> <ul style="list-style-type: none"> • Less server interaction – You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server. • Immediate feedback to the visitors – They don't have to wait for a page reload to see if they have forgotten to enter something. • Increased interactivity – You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard. • Richer interfaces – You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.
8.	<p>Predict the need for client and server side scripting.</p> <p>Client-side scripting (embedded scripts) is code that exists inside the client's HTML page. This code will be processed on the client machine and the HTML page will NOT perform a PostBack to the web-server. Traditionally, client-side scripting is used for page navigation, data validation and formatting. The language used in this scripting is JavaScript. JavaScript is compatible and is able to run on any internet browser.</p> <p>The two main benefits of client-side scripting are:</p> <ol style="list-style-type: none"> 1. The user's actions will result in an immediate response because they don't require a trip to the server. 2. Fewer resources are used and needed on the web-server. <p>Server-side scripting is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user's (client's) request to the website. The alternative is for the web server itself to deliver a static web page.</p> <p>The client-side script executes the code to the client side which is visible to the users while a server-side script is executed in the server end which users cannot see.</p>
9.	<p>Interpret how exceptions are handled in Java script.</p>

The try...catch...finally Statement

The latest versions of JavaScript added exception handling capabilities. JavaScript implements the **try...catch...finally** construct as well as the **throw** operator to handle exceptions.

You can **catch** programmer-generated and **runtime** exceptions, but you cannot **catch** JavaScript syntax errors.

Here is the **try...catch...finally** block syntax –

```
<script type = "text/javascript">
<!--
try {
    // Code to run
    [break;]
}

catch ( e ) {
    // Code to run if an exception occurs
    [break;]
}

[ finally {
    // Code that is always executed regardless of
    // an exception occurring
}]
//-->
</script>
```

The **try** block must be followed by either exactly one **catch** block or one **finally** block (or one of both). When an exception occurs in the **try** block, the exception is placed in **e** and the

catch block is executed. The optional **finally** block executes unconditionally after try/catch.

Examples

Here is an example where we are trying to call a non-existing function which in turn is raising an exception. Let us see how it behaves without **try...catch** –

```
<html>
<head>
<script type = "text/javascript">
<!--
function myFunc() {
    var a = 100;
    alert("Value of variable a is : " + a );
}
//-->
```

```

</script>
</head>

<body>
<p>Click the following to see the result:</p>

<form>
<input type = "button" value = "Click Me" onclick = "myFunc();"/>
</form>
</body>
</html>

```

Define JavaScript statement with an example.

JavaScript Statements

Example

```

var x, y, z;      // Statement 1
x = 5;            // Statement 2
y = 6;            // Statement 3
z = x + y;        // Statement 4

```

JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

10.

```

<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Statements</h2>

```

<p>In HTML, JavaScript statements are executed by the browser.</p>

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

<script>

```
document.getElementById("demo").innerHTML = "Hello Dolly.;"
```

</script>

</body>

</html>

OUTPUT

	<p>JavaScript Statements</p> <p>In HTML, JavaScript statements are executed by the browser.</p> <p>Hello Dolly.</p>
	<p>Point out any two techniques of event programming.</p> <h2>What is an Event ?</h2> <p>JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.</p> <p>When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.</p> <p>Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.</p> <p>Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.</p> <p>Please go through this small tutorial for a better understanding HTML Event Reference. Here we will see a few examples to understand a relation between Event and JavaScript –</p> <h2>onclick Event Type</h2>
11.	<p>This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.</p> <p>Example</p> <p>Try the following example.</p> <pre><html> <head> <script type = "text/javascript"> <!-- function sayHello() { alert("Hello World") } //--> </script> </head></pre>

```

<body>
    <p>Click the following button and see result</p>
    <form>
        <input type = "button" onclick = "sayHello()" value = "Say Hello" />
    </form>
</body>
</html>

```

Example 2

```

<!doctype html>
<html>
<head>
    <script>
        function hov() {
            var e = document.getElementById('hover');
            e.style.display = 'none';
        }
    </script>
</head>
<body>
    <div id="hover" onmouseover="hov()"
        style="background-color:green;height:200px;width:200px;">
    </div>
</body>
</html>

```

OUTPUT

Before mouse is taken over green square-



Green square gets disappear after mouse is taken over it.

What is DHTML?

DHTML

12.

DHTML stands for Dynamic HTML, it is totally different from HTML. The DHTML make use of Dynamic object model to make changes in settings and also in properties and methods.

	<p>DHTML allows different scripting languages in a web page to change their variables, which enhance the effects, looks and many others functions after the whole page have been fully loaded or under a view process, or otherwise static HTML pages on the same.</p> <p>DHTML is used to create interactive and animated web pages that are generated in real-time, also known as dynamic web pages so that when such a page is accessed, the code within the page is analyzed on the web server and the resulting HTML is sent to the client's web browser.</p>
13.	<p>Differentiate HTML and DHTML</p> <p>Some differences between HTML and DHTML:</p> <ul style="list-style-type: none"> • HTML is a mark-up language, while DHTML is a collection of technology. • DHTML creates dynamic web pages, whereas HTML creates static web pages. • DHTML allows including small animations and dynamic menus in Web pages. • DHML used events, methods, properties to insulate dynamism in HTML Pages. • DHML is basically using JavaScript and style sheets in an HTML page. • HTML sites will be slow upon client-side technologies, while DHTML sites will be fast enough upon client-side technologies. • HTML creates a plain page without any styles and Scripts called as HTML. Whereas, DHTML creates a page with HTML, CSS, DOM and Scripts called as DHTML. • HTML cannot have any server side code but DHTML may contain server side code. • In HTML, there is no need for database connectivity, but DHTML may require connecting to a database as it interacts with user. • HTML files are stored with .htm or .html extension, while DHTML files are stored with .dhtm extension. • HTML does not require any processing from browser, while DHTML requires processing from browser which changes its look and feel.
14.	<p>Point out the key features of DHTML.</p> <p>Key Features: Following are the some major key features of DHTML:</p> <ul style="list-style-type: none"> • Tags and their properties can be changed using DHTML. • It is used for real-time positioning. • Dynamic fonts can be generated using DHTML. • It is also used for data binding. • It makes a webpage dynamic and be used to create animations, games, applications along with providing new ways of navigating through websites. • The functionality of a webpage is enhanced due to the usage of low-bandwidth effect by DHTML. • DHTML also facilitates the use of methods, events, properties, and codes.
15.	<p>Classify the data types in JSON</p> <p>JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. ... JSON is built on two structures: A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.</p>

	<p>JSON Data Types. At the granular level, JSON consist of 6 data types. First four data types (string, number, boolean and null) can be referred as simple data types. Other two data types (object and array) can be referred as complex data types.</p>
	<p>How to convert text into a JavaScript object using JSON?</p> <h2><u>Converting JSON text to JavaScript Object</u></h2> <p>JSON (JavaScript Object Notation) is a lightweight data-interchange format. As its name suggests, JSON is derived from the JavaScript programming language, but it's available for use by many languages including Python, Ruby, PHP, and Java and hence, it can be said as language-independent. For humans, it is easy to read and write and for machines, it is easy to parse and generate. It is very useful for storing and exchanging data.</p> <p>A JSON object is a key-value data format that is typically rendered in curly braces. JSON object consist of curly braces ({ }) at the either ends and have key-value pairs inside the braces. Each key-value pair inside braces are separated by comma (,). JSON object looks something like this :</p> <pre>{ "key": "value", "key": "value", "key": "value", }</pre> <p>16. Example for a JSON object :</p> <pre>{ "rollno": 101, "name": "Mayank", "age": 20, }</pre> <h3>Conversion of JSON text to Javascript Object</h3> <p>JSON text/object can be converted into Javascript object using the function JSON.parse().</p> <pre>var object1 = JSON.parse('{"rollno":101, "name":"Mayank", "age":20}');</pre> <p>For getting the value of any key from a Javascript object, we can use the values as: object1.rollno</p>
17.	Evaluate the syntax to create arrays in JSON.

JavaScript | JSON Arrays

The JSON Arrays is similar to JavaScript Arrays.

Syntax of Arrays in JSON Objects:

```
// JSON Arrays Syntax
```

```
{  
    "name": "Peter parker",  
    "heroName": "Spiderman",  
    "friends" : ["Deadpool", "Hulk", "Wolverine"]  
}
```

Accessing Array Values:

The Array values can be accessed using the index of each element in an Array.

EXAMPLE

```
<script>  
var myObj, i, x = "";  
myObj = {  
    "name": "John",  
    "age": 30,  
    "cars": [ "Ford", "BMW", "Fiat" ]  
};  
  
for (i in myObj.cars) {  
    x += myObj.cars[i] + "<br>";  
}  
document.getElementById("demo").innerHTML = x;  
</script>
```

OUTPUT

Looping through an array using a for in loop:

Ford
BMW
Fiat

18.	<p>How will you make a request with JSON?</p> <p>What is a JSON request?</p> <p>JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).</p> <p><u>jQuery getJSON() Method</u></p>
-----	---

The jQuery getJSON() method sends asynchronous http GET request to the server and retrieves the data in JSON format by setting accepts header to application/json, text/javascript. This is same as get() method, the only difference is that getJSON() method specifically retrieves JSON data whereas get() method retrieves any type of data. It is like shortcut method to retrieve JSON data.

```
$.getJSON(url,[data],[callback]);
```

Parameter Description:

- url: request url from which you want to retrieve the data
- data: JSON data to be sent to the server as a query string
- callback: function to be executed when request succeeds

The following example shows how to retrieve JSON data using getJSON() method.

Example: jQuery getJSON() Method

```
$.getJSON('/jquery/getjsondata', {name:'Steve'}, function (data, textStatus, jqXHR){  
    $('p').append(data.firstName);  
});  
  
<p></p>
```

OUTPUT

jQuery get() method demo

Steve

Define DDL and DML

DDL(Data Definition Language) : DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

Examples of DDL commands:

- **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- **DROP** – is used to delete objects from the database.
- **ALTER**-is used to alter the structure of the database.
- **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT** –is used to add comments to the data dictionary.
- **RENAME** –is used to rename an object existing in the database

19. Write SQL query to find minimum and maximum marks in a table.

DML(Data Manipulation Language) : The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

20.

	<p>Examples of DML:</p> <ul style="list-style-type: none"> • INSERT – is used to insert data into a table. • UPDATE – is used to update existing data within a table. • DELETE – is used to delete records from a database table. 		
	<p>PART - B</p>		
Q.No	<p>Questions</p> <p>i) Examine variables and data types in JavaScript.</p> <p style="text-align: center;">Variables in JavaScript:</p> <p>Variables in JavaScript are containers which hold reusable data. It is the basic unit of storage in a program.</p> <ul style="list-style-type: none"> • The value stored in a variable can be changed during program execution. • A variable is only a name given to a memory location, all the operations done on the variable effects that memory location. • In JavaScript, all the variables must be declared before they can be used. <p>JavaScript variables are containers for storing data values.</p> <p>In this example, <code>x</code>, <code>y</code>, and <code>z</code>, are variables:</p> <p>Examples</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 10px; vertical-align: top;"> <pre>var x = 5; var y = 6; var z = x + y;</pre> </td><td style="padding: 10px; vertical-align: top;"> <pre>var price1 = 5; var price2 = 6; var total = price1 + price2;</pre> </td></tr> </table> <p>1.</p> <h2>Declaring (Creating) JavaScript Variables</h2> <p>Creating a variable in JavaScript is called "declaring" a variable.</p> <p>You declare a JavaScript variable with the <code>var</code> keyword:</p> <pre>var carName;</pre> <p>After the declaration, the variable has no value (technically it has the value of <code>undefined</code>).</p> <p>To assign a value to the variable, use the equal sign:</p> <pre>carName = "Volvo";</pre> <p>You can also assign a value to the variable when you declare it:</p> <pre>var carName = "Volvo";</pre>	<pre>var x = 5; var y = 6; var z = x + y;</pre>	<pre>var price1 = 5; var price2 = 6; var total = price1 + price2;</pre>
<pre>var x = 5; var y = 6; var z = x + y;</pre>	<pre>var price1 = 5; var price2 = 6; var total = price1 + price2;</pre>		

EXAMPLE

```
<script>
var carName = "Volvo";
document.getElementById("demo").innerHTML
= carName;
</script>
```

OUTPUT

JavaScript Variables

Create a variable, assign a value to it, and display it:

Volvo

- ii) Give various Operators in JavaScript.

JavaScript Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description	Example
+	Addition	<code>var x = 5; var y = 2; var z = x + y;</code>
-	Subtraction	<code>var x = 5; var y = 2; var z = x - y;</code>
*	Multiplication	<code>var x = 5; var y = 2; var z = x * y;</code>
**	Exponentiation (ES2016)	
/	Division	
%	Modulus (Division Remainder)	
++	Increment	
--	Decrement	

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
----------	---------	---------

=	<code>x = y</code>	<code>x = y</code>
<code>+=</code>	<code>x += y</code>	<code>x = x + y</code>
<code>-=</code>	<code>x -= y</code>	<code>x = x - y</code>
<code>*=</code>	<code>x *= y</code>	<code>x = x * y</code>
<code>/=</code>	<code>x /= y</code>	<code>x = x / y</code>
<code>%=</code>	<code>x %= y</code>	<code>x = x % y</code>
<code>**=</code>	<code>x **= y</code>	<code>x = x ** y</code>

The **addition assignment** operator (`+=`) adds a value to a variable.

Assignment

```
var x = 10;
x += 5;
```

JavaScript String Operators

The `+` operator can also be used to add (concatenate) strings.

Example

```
var txt1 = "John";
var txt2 = "Doe";
var txt3 = txt1 + " " + txt2;
```

The result of `txt3` will be:

John Doe

The `+=` assignment operator can also be used to add (concatenate) strings:

Example

```
var txt1 = "What a very ";
txt1 += "nice day";
```

The result of `txt1` will be:

What a very nice day

When used on strings, the `+` operator is called the **concatenation operator**.

JavaScript Comparison Operators

Operator	Description
<code>==</code>	equal to
<code>====</code>	equal value and equal type
<code>!=</code>	not equal
<code>!==</code>	not equal value or not equal type

>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	Description
&&	logical and
	logical or
!	logical not

JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

Type operators are fully described in the [JS Type Conversion](#) chapter.

JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

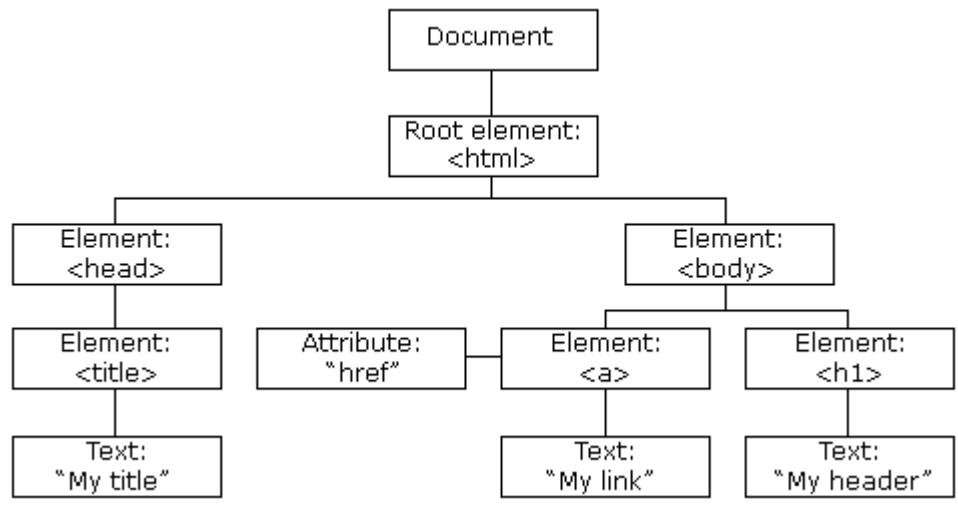
Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
	OR	5 1	0101 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2

	>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2
--	-----	-----------------------	---------	------------	------	---

The examples above uses 4 bits unsigned examples. But JavaScript uses 32-bit signed numbers. Because of this, in JavaScript, ~5 will not return 10. It will return -6.

~0000000000000000000000000000000101 will return 111111111111111111111111010

2.	<p>(i) Summarize about DOM Model.</p> <p>What is the DOM?</p> <p>The DOM is a W3C (World Wide Web Consortium) standard.</p> <p>The DOM defines a standard for accessing documents:</p> <p><i>"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."</i></p> <p>The W3C DOM standard is separated into 3 different parts:</p> <ul style="list-style-type: none"> • Core DOM - standard model for all document types • XML DOM - standard model for XML documents • HTML DOM - standard model for HTML documents <p><u>JavaScript HTML DOM</u></p> <p>With the HTML DOM, JavaScript can access and change all the elements of an HTML document.</p> <p>The HTML DOM (Document Object Model)</p> <p>When a web page is loaded, the browser creates a Document Object Model of the page.</p> <p>The HTML DOM model is constructed as a tree of Objects:</p> <p><u>The HTML DOM Tree of Objects</u></p>
----	--



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

(ii) **Describe** the concepts of Popup Boxes.

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box	Syntax	Example
<p>An alert box is often used if you want to make sure information comes through to the user.</p> <p>When an alert box pops up, the user will have to click "OK" to proceed</p>	<p>window.alert("sometext");</p> <p>The <code>window.alert()</code> method can be written without the window prefix</p>	<pre>alert("I am an alert box!");</pre>

	<p>Confirm Box</p> <p>When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.</p> <p>If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.</p>	<p>Syntax</p> <pre>window.confirm("sometext");</pre> <p>The window.confirm() method can be written without the window prefix.</p>	<p>Example</p> <pre>if(confirm("Press a button!")) { txt = "You pressed OK!"; } else { txt = "You pressed Cancel!"; }</pre>
	<p>Prompt Box</p> <p>A prompt box is often used if you want the user to input a value before entering a page.</p> <p>If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.</p>	<p>Syntax</p> <pre>window.prompt("sometext","defaultText");</pre> <p>The window.prompt() method can be written without the window prefix.</p>	<p>Example</p> <pre>var person = prompt("Please enter your name", "Harry Potter"); if(person == null person == "") { txt = "User cancelled the prompt."; } else { txt = "Hello " + person + "! How are you today?"; }</pre>
3.	<p>(i) Write short notes on Date and Objects.</p> <p>The Date Object. The Date object is a built-in object in JavaScript that stores the date and time. It provides a number of built-in methods for formatting and managing that data. By default, a new Date instance without arguments provided creates an object corresponding to the current date and time</p> <h3><u>The Date Object</u></h3> <p>The Date object is a built-in object in JavaScript that stores the date and time. It provides a number of built-in methods for formatting and managing that data.</p> <p>By default, a new Date instance without arguments provided creates an object corresponding to the current date and time. To demonstrate JavaScript's Date, let's create a variable and assign the current date to it.</p>		

EXAMPLE

now.js

```
// Set variable to current date and time
const now = new Date();

// View the output
now;
Output
Wed Oct 18 2017 12:41:34 GMT+0000 (UTC)
```

Date Creation	Output
<code>new Date()</code>	Current date and time
<code>new Date(timestamp)</code>	Creates date based on milliseconds since Epoch time
<code>new Date(date string)</code>	Creates date based on date string
<code>new Date(year, month, day, hours, minutes, seconds, milliseconds)</code>	Creates date based on specified date and time

(ii) **Explain** in detail about Regular expressions.

A JavaScript Regular Expression (or **Regex**) is a sequence of characters that we can utilize to work effectively with strings. Using this syntax, we can:

- **search** for text in a string
- **replace** substrings in a string
- **extract** information from a string

4.	<p>(i) Describe the control statements in Java.</p> <p>Conditional Statements</p> <p>Very often when you write code, you want to perform different actions for different decisions.</p>
----	---

You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

JavaScript Loops

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

```
<script>

var cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];

var text = "";

var i;
for (i = 0; i < cars.length; i++) {
    text += cars[i] + "<br>";

}

document.getElementById("demo").innerHTML = text;

</script>
```

OUTPUT

JavaScript For Loop

BMW
Volvo
Saab
Ford
Fiat
Audi

	(ii) Discuss any two validation functions in java script.
5.	<p>(i) Write a Java script to find the Prime number between 1 and 100.</p> <pre><html> <head> <title>JavaScript Prime</title> </head> <body> <script> for (var limit = 1; limit <= 20; limit++) { var a = false; for (var i = 2; i <= limit; i++) { if (limit%i==0 && i!=limit) { a = true; } } if (a === false) { document.write("
" + limit); } } </script> </body> </html></pre> <p>(ii) Write a Java Script to find factorial of a given number.</p> <pre>function factorial(n) { return (n != 1) ? n * factorial(n - 1) : 1; } alert(factorial(5)); // 120</pre>
6.	<p>(i) Demonstrate a java script for displaying the context menu.</p> <p>(ii) Demonstrate a java script to display the welcome message using the alert whenever button of a html form is pressed.</p> <pre><html> <head> <title>Display Alert Message on Button Click Event.</title> <script type="text/javascript"> function showMessage(){ alert("Welcome friends, You pressed the Button."); } </script> </head> <body></pre>

```

<center>
    <h1>Display Alert Message on Button Click Event.</h1>
    <b>Click on button to display message:</b><br><br>
    <input type="button" id="btnShowMsg" value="Click Me!">
    onClick='showMessage()'
</center>
</body>
</html>

```

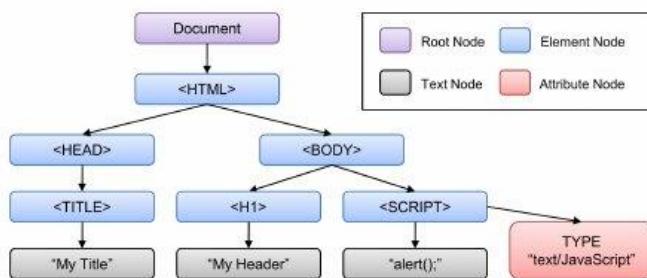
Result

Welcome friends, You pressed the Button.

- (i) Evaluate how DOM Nodes and Trees can be used.

What is the purpose of HTML DOM Node Tree?

- HTML DOM view the HTML document with a tree structure format and it consists of root node and child nodes.
- The node-tree is being accessed using the tree formation and the structure in which the elements get created.
- The contents that are used being modified or removed using the new elements and it can be created within the limitations.
- The structure consists of a document that is the root and within it Root element <html> from where the tree starts.
- It consists of sub-elements like <head> and <body> and other text and attributes written in the HTML format.



- (ii) Evaluate the way of Traversing and modifying a DOM Tree.

8. (i) **Discuss the concepts of Registering Event handlers.**
As mentioned above, **events** are actions or occurrences that happen in the system you are programming — the system produces (or "fires") a signal of some kind when an event occurs,

and also provides a mechanism by which some kind of action can be automatically taken (that is, some code running) when the event occurs. For example in an airport when the runway is clear for a plane to take off, a signal is communicated to the pilot, and as a result, they commence piloting the plane.

In the case of the Web, events are fired inside the browser window, and tend to be attached to a specific item that resides in it — this might be a single element, set of elements, the HTML document loaded in the current tab, or the entire browser window. There are a lot of different types of events that can occur, for example:

The user clicking the mouse over a certain element or hovering the cursor over a certain element.

The user pressing a key on the keyboard.

The user resizing or closing the browser window.

A web page finishing loading.

A form being submitted.

A video being played, or paused, or finishing play.

An error occurring.

Each available event has an **event handler**, which is a block of code (usually a JavaScript function that you as a programmer create) that will be run when the event fires. When such a block of code is defined to be run in response to an event firing, we say we are **registering an event handler**. Note that event handlers are sometimes called **event listeners** — they are pretty much interchangeable for our purposes, although strictly speaking, they work together. The listener listens out for the event happening, and the handler is the code that is run in response to it happening.

In the following example, we have a single `<button>`, which when pressed, makes the background change to a random color:

```
<button>Change color</button>
```

The JavaScript looks like so:

```
const btn = document.querySelector('button');

function random(number) {
    return Math.floor(Math.random() * (number+1));
}

btn.onclick = function() {
    const rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' +
random(255) + ')';
    document.body.style.backgroundColor = rndCol;
}
```

(ii) **Discuss** the concepts of Event Handling.

JavaScript Events

HTML events are "**things**" that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can "**react**" on these events.

HTML Events

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

With single quotes:

`<element event='some JavaScript'>`

With double quotes:

`<element event="some JavaScript">`

In the following example, an **onclick** attribute (with code), is added to a `<button>` element:

Example

```
<button onclick="document.getElementById('demo').innerHTML = Date()">The time  
is?</button>
```

OUTPUT

Sun Jan 12 2020 06:00:41 GMT+0530 (India Standard Time)

In the example above, the JavaScript code changes the content of the element with id="demo".

9. **Analyze** a web page to create a clock with timing event.
- ```
function showTime(){
 var date = new Date();
 var h = date.getHours(); // 0 - 23
 var m = date.getMinutes(); // 0 - 59
```

```
var s = date.getSeconds(); // 0 - 59
var session = "AM";

if(h == 0){
 h = 12;
}

if(h > 12){
 h = h - 12;
 session = "PM";
}

h = (h < 10) ? "0" + h : h;
m = (m < 10) ? "0" + m : m;
s = (s < 10) ? "0" + s : s;

var time = h + ":" + m + ":" + s + " " + session;
document.getElementById("MyClockDisplay").innerText = time;
document.getElementById("MyClockDisplay").textContent = time;

setTimeout(showTime, 1000);

}

showTime();
```

---

```
<html>
<head>
<title>Digital Clock</title>
<style>
#clock{
 color:#F0F;
}
</style>
</head>
<body>
<script>
function digclock()
{
 var d = new Date()
 var t = d.toLocaleTimeString()
```

```

 document.getElementById("clock").innerHTML = t
 }

 setInterval(function(){digclock()},1000)

</script>
Digital Clock
<div id="clock">

</div>
</body>
</html>

```

**OUTPUT**

**Output :**

Digital Clock  
**5:54:26 PM**

10. (i) **Write short notes on DHTML with JavaScript.**
- Dynamic HyperText Markup Language (DHTML) is a combination of Web development technologies used to create dynamically changing websites. Web pages may include animation, dynamic menus and text effects. The technologies used include a combination of HTML, JavaScript or VB Script, CSS and the document object model (DOM).
- Designed to enhance a Web user's experience, DHTML includes the following features:
- Dynamic content, which allows the user to dynamically change Web page content
  - Dynamic positioning of Web page elements
  - Dynamic style, which allows the user to change the Web page's color, font, size or content
- EXAMPLE**
- ```

<!DOCTYPE html PUBLIC "-//abc//DTD XHTML 1.1//EN"
"http://www.abc.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.abc.org/1999/xhtml">
<head>
<title>DHTML example</title>
<script type="text/JavaScript">
    function greet_user()
    {
        var name=document.getElementById("userName").value;

```

```

if(name=="")
{
    alert("Welcome"+name);
}
else
{
    alert("Please provide User Name")
}

```

</script>

</head>

<body>

| | |
|--|--|
| <h6> Please Enter Your Name </h6></td> | |
| <td><h4>User Name </h4></td> <td><input type="text" id="userName" ></td> | |
| <td colspan="2"><input type="button" value="Submit" onclick="greet_user()"/> | |

(ii) **Classify** moving elements.

Explain the concept of JSON with example.

JSON is a format for storing and transporting data.

JSON is often used when data is sent from a server to a web page.

What is JSON?

11.

- JSON stands for **JavaScripT Object Notation**
- JSON is a lightweight data interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand

* The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language.

JSON Example

This JSON syntax defines an employees object: an array of 3 employee records (objects):

JSON Example

```
{  
  "employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
  ]  
}
```

Describe in detail about JSON Objects and Arrays.

JSON Arrays

Arrays as JSON Objects

Example

```
[ "Ford", "BMW", "Fiat" ]
```

Arrays in JSON are almost the same as arrays in JavaScript.

In JSON, array values must be of type string, number, object, array, boolean or *null*.

In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and *undefined*.

12.

Arrays in JSON Objects

Arrays can be values of an object property:

Example

```
{  
  "name": "John",  
  "age": 30,  
  "cars": [ "Ford", "BMW", "Fiat" ]  
}
```

Accessing Array Values

You access the array values by using the index number:

Example

```
x = myObj.cars[0];
```

Looping Through an Array

You can access array values by using a **for-in** loop:

Example

```
for (i in myObj.cars) {  
    x += myObj.cars[i];  
}
```

OUTPUT

Looping through an array using a for in loop:

```
Ford  
BMW  
Fiat
```

Analyze about Function files and Http Request with sample program.

```
<!doctype html>  
<title>Example</title>  
  
<script>  
// Store XMLHttpRequest and the JSON file location in variables  
var xhr = new XMLHttpRequest();  
var url = "https://www.quackit.com/json/tutorial/artists.txt";  
  
// Called whenever the readyState attribute changes  
xhr.onreadystatechange = function() {  
  
    // Check if fetch request is done  
    if (xhr.readyState == 4 && xhr.status == 200) {  
  
        // Parse the JSON string  
        var jsonData = JSON.parse(xhr.responseText);  
  
        // Call the showArtists(), passing in the parsed JSON string  
        showArtists(jsonData);  
    }  
};
```

```

// Do the HTTP call using the url variable we specified above
xhr.open("GET", url, true);
xhr.send();

// Function that formats the string with HTML tags, then outputs the result
function showArtists(data) {
    var output = "<ul>"; // Open list
    var i;

    // Loop through the artists, and add them as list items
    for (var i in data.artists) {
        output += "<li>" + data.artists[i].artistname + " (Born: " + data.artists[i].born + ")</li>";
    }

    output += "</ul>"; // Close list

    // Output the data to the "artistlist" element
    document.getElementById("artistList").innerHTML = output;
}

</script>

<!-- The output appears here -->
<div id="artistList"></div>

```

OUTPUT

- Leonard Cohen (Born: 1934)
- Joe Satriani (Born: 1956)
- Snoop Dogg (Born: 1971)

| | |
|-----|---|
| 14. | <p>Summarize about</p> <p>(i) SQL Data Definition Commands</p> <p>Examples of Sql DDL commands are</p> <ul style="list-style-type: none"> • CREATE – Create an object. I mean, <u>create a database</u>, <u>table</u>, <u>triggers</u>, index, <u>functions</u>, <u>stored procedures</u>, etc. • DROP – This SQL DDL command helps to delete objects. For example, delete tables, delete a database, etc. • <u>ALTER</u> – Used to alter the existing database or its object structures. • <u>TRUNCATE</u> – This SQL DDL command removes records from tables • <u>RENAME</u> – Renaming the database objects <p>(ii) Data Manipulation Commands.</p> <p>Examples of DML commands are</p> <ul style="list-style-type: none"> • <u>SELECT</u> – This SQL DML command select records or data from a table • <u>INSERT</u> – Insert data into a database table. • <u>UPDATE</u> – This SQL DML command will update existing records within a table |
|-----|---|

| | |
|-----------------|--|
| | <ul style="list-style-type: none"> • <u>DELETE</u> – Delete unwanted records from a table |
| PART – C | |
| Q.No | Questions |
| 1. | <p>Analyze the merits and demerits of DOM parser with neat example.</p> <pre> <!DOCTYPE html> <html> <head> <script type="text/javascript"> var str = "<root><customer name='John' address='Chicago'></customer></root>"; function CreateXMLDocument () { var xmlDoc = null; if (window.DOMParser) { var parser = new DOMParser (); xmlDoc = parser.parseFromString (str, "text/xml"); } else if (window.ActiveXObject) { xmlDoc = new ActiveXObject ("Microsoft.XMLDOM"); xmlDoc.async = false; xmlDoc.loadXML (str); } var customerNode = xmlDoc.getElementsByTagName ("customer")[0]; var customerName = customerNode.getAttribute ("name"); alert ("The name of the first customer is " + customerName); } </script> </head> <body> <button onclick="CreateXMLDocument ();">Create an XML document from a string</button> </body> </html></pre> |
| 2. | <p>Consider a java script and HTML to create a page with two panes. The first pane (on left) should have a text area where HTML code can be typed by the user. The pane on the left should have a text area where HTML code can be typed by the user. The pane on the right side should display the preview of the HTML code typed by the user, as it would be seen on the browser.</p> |
| 3. | <p>Develop a DHTML program to handle the user click event.</p> <pre> <!DOCTYPE html> <html></pre> |

| | |
|----|---|
| | <pre><body> <p>Click the button to display the time.</p> <button onclick="getElementById('demo').innerHTML=Date()">What is the time?</button> <p id="demo"></p> </body> </html></pre> <p>OUTPUT</p> <p>Click the button to display the time.</p> <p>What is the time?</p> <p>Sat Jan 11 2020 22:36:24 GMT+0530 (India Standard Time)</p> |
| 4. | <p>Formulate a JavaScript program that work with JSON.</p> <p>JSON Object Syntax</p> <p>Example</p> <pre>{ "name":"John", "age":30, "car":null }</pre> <ul style="list-style-type: none"> ➤ JSON objects are surrounded by curly braces {}. ➤ JSON objects are written in key/value pairs. ➤ Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null). ➤ Keys and values are separated by a colon. ➤ Each key/value pair is separated by a comma. <p>Accessing Object Values</p> <p>You can access the object values by using dot (.) notation:</p> <p>Example</p> <pre>myObj = { "name":"John", "age":30, "car":null }; x = myObj.name;</pre> <p>You can also access the object values by using bracket ([]) notation:</p> <p>Example</p> |

```
myObj = { "name":"John", "age":30, "car":null };
x = myObj["name"];
```

Looping an Object

You can loop through object properties by using the for-in loop:

Example

In a for-in loop, use the bracket notation to access the property *values*:

Example

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

<p>How to loop through all properties in a JSON object.</p>

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

<script>
var myObj, x;
myObj = {"name":"John", "age":30, "car":null};
for (x in myObj) {
  document.getElementById("demo").innerHTML += x + " " + myObj[x] + "<br>";
}
</script>

</body>
</html>
```

OUTPUT

How to loop through all properties in a JSON object.

```
name John
age 30
car null
```

Nested JSON Objects

Values in a JSON object can be another JSON object.

Example

```
myObj = {
  "name":"John",
```

```
"age":30,  
"cars": {  
    "car1":"Ford",  
    "car2":"BMW",  
    "car3":"Fiat"  
}  
}
```

You can access nested JSON objects by using the dot notation or bracket notation:

Example

```
x = myObj.cars.car2;  
// or:  
x = myObj.cars["car2"];
```

OUTPUT

How to access nested JSON objects.

BMW
BMW

CS8651 Internet Programming – 2017Reg

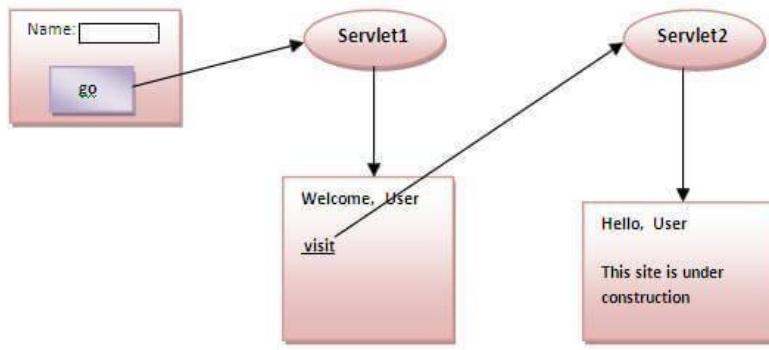
**UNIT III
- SERVER SIDE PROGRAMMING**

Servlets: Java Servlet Architecture – Servlet Life Cycle – Form GET and POST actions – Session Handling – Understanding Cookies – Installing and Configuring Apache Tomcat Web Server; **DATABASE CONNECTIVITY:** JDBC perspectives, JDBC program example; **JSP:** Understanding Java Server Pages – JSP Standard Tag Library (JSTL) – Creating HTML forms by embedding JSP code.

PART-A

Q.No	Questions
1.	<p>What are servlets?</p> <p>A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.</p> <ul style="list-style-type: none"> ➤ Servlet is a technology which is used to create a web application. ➤ Servlet is an API that provides many interfaces and classes including documentation. ➤ Servlet is an interface that must be implemented for creating any Servlet. ➤ Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests. ➤ Servlet is a web component that is deployed on the server to create a dynamic web page.
2.	<p>List the application of servlets.</p> <p>Servlets may be used at different levels on a distributed framework. The following are some examples of servlet usage:</p> <ul style="list-style-type: none"> ➤ To accept form input and generate HTML Web pages dynamically.

	<ul style="list-style-type: none"> ➤ As part of middle tiers in enterprise networks by connecting to SQL databases via JDBC. ➤ In conjunction with applets to provide a high degree of interactivity and dynamic Web content generation. ➤ For collaborative applications such as online conferencing. ➤ A community of servlets could act as active agents which share data with each other. ➤ Servlets could be used for balancing load among servers which mirror the same content. ➤ Protocol support is one of the most viable uses for servlets. For example, a file service can start with NFS and move on to as many protocols as desired; the transfer between the protocols would be made transparent by servlets. Servlets could be used for tunneling over HTTP to provide chat, newsgroup or other file server functions.
3.	<p>Summarize the advantages and disadvantages of servlets.</p> <p>Servlet Advantage</p> <ul style="list-style-type: none"> ➤ Servlets provide a way to generate dynamic documents that is both easier to write and faster to run. ➤ Provide all the powerfull features of JAVA, such as Exception handling and garbage collection. ➤ Servlet enables easy portability across Web Servers. ➤ Servlet can communicate with different servlet and servers. ➤ Since all web applications are stateless protocol, servlet uses its own API to maintain session <p>Servlet Disadvantage</p> <ul style="list-style-type: none"> ➤ Designing in servlet is difficult and slows down the application. ➤ Writing complex business logic makes the application difficult to understand. ➤ You need a Java Runtime Environment on the server to run servlets. ➤ CGI is a completely language independent protocol, so you can write CGIs in whatever languages you have available (including Java if you want to).
4.	<p>Show how is session tracking is achieved by the URL rewriting?</p> <p>URL Rewriting</p> <p>In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format:</p> <p>url?name1=value1&name2=value2&??</p> <p>A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.</p>



Advantage of URL Rewriting

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

1. It will work only with links.
2. It can send Only textual information.

URL Rewriting

URL rewriting is another way to support anonymous session tracking. With URL rewriting, every local URL the user might click on is dynamically modified, or rewritten, to include extra information. The extra information can be in the form of extra path information, added parameters, or some custom, server-specific URL change.

Example 7-2 shows a revised version of our shopping cart viewer that uses URL rewriting in the form of extra path information to anonymously track a shopping cart.

```

Example 7-2. Session tracking using URL rewriting
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ShoppingCartViewerRewrite extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
        ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("");
        out.println("");
        // Get the current session ID, or generate one if necessary
        String sessionid = req.getPathInfo();
        if (sessionid == null)
            sessionid = generateSessionId();
    }
}

```

```

}

// Cart items are associated with the session ID
String[] items = getItemsFromCart(sessionid);
// Print the current cart items.
out.println("You currently have the following items in your cart:
");
if (items == null) { out.println("None"); } else { out.println(
    );
    for (int i = 0; i < items.length; i++)
    { out.println(
        • " + items[i]); } out.println(
    );
}

// Ask if the user wants to add more items or check out. // Include the session
ID in the action URL.
out.println(
");
out.println("Would you like to
");
out.println(" \\""); out.println(" \\""); out.println(
");

// Offer a help page. Include the session ID in the URL. out.println("For
help, click here");
out.println(""); }

private static String generateSessionId() { String uid = new
java.rmi.server.UID().toString();

// guaranteed unique return java.net.URLEncoder.encode(uid);

// encode any special chars }

private static String[] getItemsFromCart(String sessionid) {

// Not implemented }

}

This servlet first tries to retrieve the current session ID using
getPathInfo() . If a session ID is not specified, it calls
generateSessionId() to generate a new unique session ID using an RMI

```

	<p>class designed specifically for this. The session ID is used to fetch and display the current items in the cart. The ID is then added to the form's ACTION attribute, so it can be retrieved by the ShoppingCart servlet. The session ID is also added to a new help URL that invokes the Help servlet. This wasn't possible with hidden form fields because the Help servlet isn't the target of a form submission. docstore.mik.ua/oreilly/javaent/servlet/ch07_03.htm</p>												
5.	<p>Compare GET and POST request type.</p> <table border="1"> <thead> <tr> <th>GET</th><th>POST</th></tr> </thead> <tbody> <tr> <td>1) In case of Get request, only limited amount of data can be sent because data is sent in header.</td><td>In case of post request, large amount of data can be sent because data is sent in body.</td></tr> <tr> <td>2) Get request is not secured because data is exposed in URL bar.</td><td>Post request is secured because data is not exposed in URL bar.</td></tr> <tr> <td>3) Get request can be bookmarked.</td><td>Post request cannot be bookmarked.</td></tr> <tr> <td>4) Get request is idempotent. It means second request will be ignored until response of first request is delivered</td><td>Post request is non-idempotent.</td></tr> <tr> <td>5) Get request is more efficient and used more than Post.</td><td>Post request is less efficient and used less than get.</td></tr> </tbody> </table>	GET	POST	1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.	2) Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.	3) Get request can be bookmarked.	Post request cannot be bookmarked.	4) Get request is idempotent . It means second request will be ignored until response of first request is delivered	Post request is non-idempotent .	5) Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.
GET	POST												
1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.												
2) Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.												
3) Get request can be bookmarked.	Post request cannot be bookmarked.												
4) Get request is idempotent . It means second request will be ignored until response of first request is delivered	Post request is non-idempotent .												
5) Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.												
6.	<p>Summarize the servlet interface and its methods.</p> <p>Servlet Interface</p> <ol style="list-style-type: none"> 1. Servlet Interface 2. Methods of Servlet interface <ul style="list-style-type: none"> ➤ Servlet interface provides common behavior to all the servlets. ➤ Servlet interface defines methods that all servlets must implement. 												

- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).
- It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

Servlet Example by implementing Servlet interface

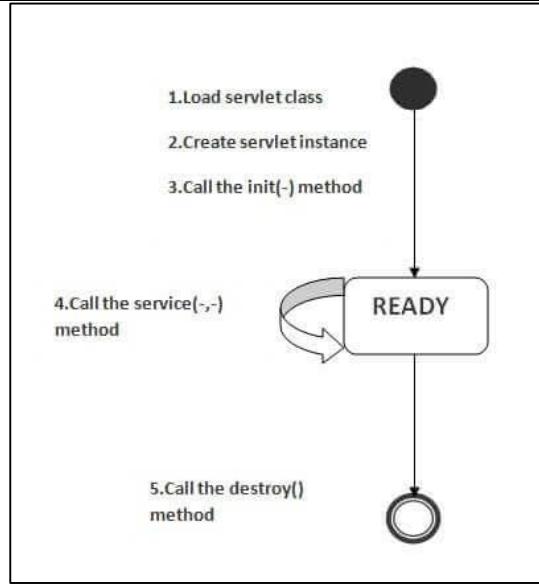
File: First.java

```

1. import java.io.*;
2. import javax.servlet.*;
3.
4. public class First implements Servlet{
5.     ServletConfig config=null;
6.
7.     public void init(ServletConfig config){
8.         this.config=config;
9.         System.out.println("servlet is initialized");
10.    }

```

	<pre> 11. 12. public void service(ServletRequest req,ServletResponse res) 13. throws IOException,ServletException{ 14. 15. res.setContentType("text/html"); 16. 17. PrintWriter out=res.getWriter(); 18. out.print("<html><body>"); 19. out.print("hello simple servlet"); 20. out.print("</body></html>"); 21. 22. } 23. public void destroy(){System.out.println("servlet is destroyed");} 24. public ServletConfig getServletConfig(){return config;} 25. public String getServletInfo(){return "copyright 2007-1010";} 26. 27. }</pre>
7.	<p>Sketch the Servlet life cycle.</p> <h3>Life Cycle of a Servlet (Servlet Life Cycle)</h3> <p>The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:</p> <ol style="list-style-type: none"> A. <u>Life Cycle of a Servlet</u> <ol style="list-style-type: none"> 1. <u>Servlet class is loaded</u> 2. <u>Servlet instance is created</u> 3. <u>init method is invoked</u> 4. <u>service method is invoked</u> 5. <u>destroy method is invoked</u>



As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

Show the use of 'param' variable in JSP.

Jsp param

<jsp:param> tag is used to represent parameter value during jsp forward or include action this should be the sub tag of **<jsp:forward>** or **<jsp:include>**.

When an include or forward element is invoked, the original request object is provided to the target page. If you wish to provide additional data to that page, you can append parameters to the request object by using the **jsp:param** element.

8.

Syntax

```
<jsp:param name=" " value=" "/>
```

Example

```
<jsp:include page="contact.jsp"/>
<jsp:param name="param1" value="value1"/>
</jsp:include>
```

Example

```
<jsp:forward page="home.jsp"/>
```

	<pre><jsp:param name="param1" value="value1"/> </jsp:forward></pre>
	<p>Quote the uses of cookies.</p> <p><u>Cookies in Servlet</u></p> <p>A cookie is a small piece of information that is persisted between the multiple client requests.</p> <p>A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.</p> <p>How Cookie works</p> <p>By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.</p> <p>Types of Cookie</p> <p>There are 2 types of cookies in servlets.</p> <p>9.</p> <ol style="list-style-type: none"> 1. Non-persistent cookie 2. Persistent cookie <p>Non-persistent cookie</p> <p>It is valid for single session only. It is removed each time when user closes the browser.</p> <p>Persistent cookie</p> <p>It is valid for multiple session . It is not removed each time when user closes the browser. It is removed only if user logout or signout.</p> <p>Advantage of Cookies</p> <ol style="list-style-type: none"> 1. Simplest technique of maintaining the state. 2. Cookies are maintained at client side. <p>Disadvantage of Cookies</p> <ol style="list-style-type: none"> 1. It will not work if cookie is disabled from the browser.

	2. Only textual information can be set in Cookie object.
10.	<p>Analyze about java scriptlet.</p> <p>In JavaServer Pages (JSP) technology, a scriptlet is a piece of Java-code embedded in the HTML-like JSP code. The scriptlet is everything inside the <code><% %></code> tags. Between these the user can add any valid Scriptlet i.e. any valid Java Code. In AppleScript, a scriptlet is a small script.</p> <h3>JSP scriptlet tag</h3> <p>A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:</p> <pre><% java source code %></pre> <h3>Example of JSP scriptlet tag</h3> <p>In this example, we are displaying a welcome message.</p> <pre><html> <body> <% out.print("welcome to jsp"); %> </body> </html></pre>
11.	<p>Express appropriate java script code to remove an element (current element) from a DOM.</p> <p>A. Removing Existing HTML Elements</p> <p>To remove an HTML element, use the <code>remove()</code> method:</p> <p style="text-align: center;">1. Example</p> <pre><div> <p id="p1">This is a paragraph.</p> <p id="p2">This is another paragraph.</p> </div> <script> var elmnt = document.getElementById("p1"); elmnt.remove(); </script></pre>
12.	Define JSP.

	<p>JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.</p> <p>A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.</p> <p>Using JSP, you can collect input from users through Webpage forms, present records from a database or another source, and create Webpages dynamically.</p> <p>JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages, and sharing information between requests, pages etc.</p>
13.	<p>List any three advantages of java servlet over JSP.</p> <ol style="list-style-type: none"> 1. Being an extension to <u>Java servlet</u>, it can use every feature of Java Servlet. Also, custom tags can be used along with it. 2. There is no need to recompile JSP when changed. The changes automatically appear when run. 3. The tags which are used are easy to understand and write. 4. Supports Java API's which can now be easily used and integrated with the HTML code. 5. The results which are obtained are in HTML format, so can be opened on any browsers. 6. Customized JSP tags can be used. Ex: Tags with XML. 7. Changes can be added into the business logic page rather than changing in each and every page.
14.	<p>Rewrite the code segment to store current server time in session using Java Servlet API.</p> <p>Write a servlet application to print the current date and time.</p> <p>Answer:</p> <p>The most important advantage of using Servlet is that we can use all the</p>

methods available in core java. The Date class is available in java.util package.

Below program shows how to print the current date and time. We can use simple Date object with `toString()` to print current date and time.

DateSrv.java

```
import java.io.*;
import javax.servlet.*;

public class DateSrv extends GenericServlet
{
    //implement service()
    public void service(ServletRequest req, ServletResponse res) throws
IOException, ServletException
    {
        //set response content type
        res.setContentType("text/html");
        //get stream obj
        PrintWriter pw = res.getWriter();
        //write req processing logic
        java.util.Date date = new java.util.Date();
        pw.println("<h2>"+"Current Date & Time: "
+date.toString()+"</h2>");
        //close stream object
        pw.close();
    }
}
```

Output:

Current Date & Time: Mon Dec 12 18:01:39 IST 2016

15.

Compare the difference between JSP and servlet.

Difference Between Servlet and JSP

In this article we will list some of the differences between Servlets and JSP.

SERVLET	JSP
A servlet is a server-side program and written purely on Java.	JSP is an interface on top of Servlets. In another way, we can say that JSPs are extension of servlets to minimize the effort of developers to write User Interfaces using Java programming.
Servlets run faster than JSP	JSP runs slower because it has the transition phase for converting from JSP page to a Servlet file. Once it is converted to a Servlet then it will start the compilation
Executes inside a Web server, such as Tomcat	A JSP program is compiled into a Java servlet before execution. Once it is compiled into a servlet, its life cycle will be same as of servlet. But, JSP has its own API for the lifecycle.
Receives HTTP requests from users and provides HTTP responses	Easier to write than servlets as it is similar to HTML.
We can not build any custom tags	One of the key advantages is we can build custom tags using JSP API (there is a separate

		package available for writing the custom tags which can be available as the re-usable components with lot of flexibility
	<p>Servlet advantages include:</p> <p>1. Performance : get loaded upon first request and remains in memory idenfinitely.</p> <p>2. Simplicity : Run inside controlled server environment. No specific client software is needed:web broser is enough</p> <p>3. Session Management : overcomes HTTP's stateless nature</p> <p>4. Java Technology : network access,Database connectivity, j2ee integration</p>	<p>JSP Provides an extensive infrastructure for:</p> <ul style="list-style-type: none"> 1. Tracking sessions. 2. Managing cookies. 3. Reading and sending HTML headers. 4. Parsing and decoding HTML form data. <p>5. JSP is Efficient: Every request for a JSP is handled by a simple Java thread</p> <p>6. JSP is Scalable: Easy integration with other backend services</p> <p>7. Separation of roles: Developers, Content Authors/Graphic Designers/Web Masters</p>
<h2>II. Difference between Servlet and JSP</h2> <p>In brief, it can be defined as Servlet are the java programs that run on a Web server and act as a middle layer between a request coming from HTTP client and databases or applications on the HTTP server. While JSP is simply a text document that contains two types of text: static text which is predefined and dynamic text which is rendered after server response is received.</p>		

sr. No.	Key	Servlet	JSP
1	Implementation	Servlet is developed on Java language.	JSP is primarily written in HTML language although Java code could also be written on it but for it, JSTL or other language is required.
2	MVC	In contrast to MVC we can state servlet as a controller which receives the request process and send back the response.	On the other hand, JSP plays the role of view to render the response returned by the servlet.
3	Request type	Servlets can accept and process all type of protocol requests.	JSP on the other hand is compatible with HTTP request only.
4	Session Management	In Servlet by default session management is not enabled, the user has to enable it explicitly.	On the other hand in JSP session management is automatically enabled.
5	Performance	Servlet is faster than JSP.	JSP is slower than Servlet because first the translation of JSP to java code is taking place and then compiles.
6	Modification reflected	Modification in Servlet is a time-consuming task because it includes reloading, recompiling and restarting the server as we made any change in our code to get reflected.	On the other hands JSP modification is fast as just need to click the refresh button and code change would get reflected.

16.

Summarize briefly about the interaction between a webserver and a servlet.

How does a web server interact with a servlet?

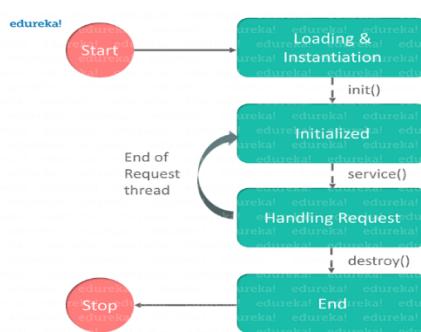
A servlet is a [Java Programming](#) language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. It is also a web component that is deployed on the server to create a dynamic web page.



In this figure you can see, a client sends a request to the server and the server generates the response, analyses it and sends the response to the client.

Stages of the Servlet Life Cycle: The Servlet life cycle mainly goes through four stages,

- Loading a Servlet.
- Initializing the Servlet.
- Request handling
- Destroying the Servlet.



	<p>Let's look at each of these stages in details:</p> <ol style="list-style-type: none"> 1. Loading a Servlet: The first stage of the Servlet life cycle involves loading and initializing the Servlet by the Servlet container. The Web container or Servlet Container can load the Servlet at either of the following two stages :Initializing the context, on configuring the Servlet with a zero or positive integer value.If the Servlet is not preceding the stage, it may delay the loading process until the Web container determines that this Servlet is needed to service a request. 2. Initializing a Servlet: After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object. The container initializes the Servlet object by invoking the <i>init(ServletConfig)</i> method which accepts ServletConfig object reference as a parameter. 3. Handling request: After initialization, the Servlet instance is ready to serve the client requests. The Servlet container performs the following operations when the Servlet instance is located to service a request :It creates the ServletRequest and ServletResponse. In this case, if this is an HTTP request then the Web container creates HttpServletRequest and HttpServletResponse objects which are subtypes of the ServletRequest and ServletResponse objects respectively. 4. Destroying a Servlet: When a Servlet container decides to destroy the Servlet, it performs the following operations,It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.After currently running threads have completed their jobs, the Servlet container calls the destroy() method on the Servlet instance. <p>After the destroy() method is executed, the Servlet container releases all the references of this Servlet instance so that it becomes eligible for garbage collection.</p>
17.	<p>Define JDBC.</p> <h2>What is JDBC?</h2> <p>JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.</p> <p>The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.</p> <ul style="list-style-type: none"> • Making a connection to a database. • Creating SQL or MySQL statements. • Executing SQL or MySQL queries in the database. • Viewing & Modifying the resulting records.

	<p>Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as –</p> <ul style="list-style-type: none"> • Java Applications • Java Applets • Java Servlets • Java ServerPages (JSPs) • Enterprise JavaBeans (EJBs). <p>All of these different executables are able to use a JDBC driver to access a database, and take advantage of the stored data.</p> <p>JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.</p>
	<p>Formulate the three methods that are central to the life cycle of the servlet.</p> <h2 style="text-align: center;">Servlets - Life Cycle</h2> <p>A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.</p> <ul style="list-style-type: none"> • The servlet is initialized by calling the init() method. • The servlet calls service() method to process a client's request. • The servlet is terminated by calling the destroy() method. • Finally, servlet is garbage collected by the garbage collector of the JVM. <p>Now let us discuss the life cycle methods in detail.</p> <h3>18. The init() Method</h3> <p>The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.</p> <p>The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.</p> <p>When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.</p> <p>The init method definition looks like this –</p> <pre style="background-color: #f0f0f0; padding: 10px;"><code>public void init() throws ServletException { // Initialization code... }</code></pre>

The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

A. The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this –

```
public void destroy() {  
    // Finalization code...  
}
```

19.	<p>Distinguish between servlets and JSP.</p> <p>Difference between Servlet and JSP</p> <table border="1" data-bbox="354 1298 1576 1983"><thead><tr><th data-bbox="354 1298 843 1365">SERVLET</th><th data-bbox="843 1298 1576 1365">JSP</th></tr></thead><tbody><tr><td data-bbox="354 1365 843 1432">Servlet is a java code.</td><td data-bbox="843 1365 1576 1432">JSP is a html based code.</td></tr><tr><td data-bbox="354 1432 843 1545">Writing code for servlet is harder than JSP as it is html in java.</td><td data-bbox="843 1432 1576 1545">JSP is easy to code as it is java in html.</td></tr><tr><td data-bbox="354 1545 843 1657">Servlet plays a controller role in MVC approach.</td><td data-bbox="843 1545 1576 1657">JSP is the view in MVC approach for showing output.</td></tr><tr><td data-bbox="354 1657 843 1792">Servlet is faster than JSP.</td><td data-bbox="843 1657 1576 1792">JSP is slower than Servlet because the first step in JSP lifecycle is the translation of JSP to java code and then compile.</td></tr><tr><td data-bbox="354 1792 843 1904">Servlet can accept all protocol requests.</td><td data-bbox="843 1792 1576 1904">JSP only accept http requests.</td></tr><tr><td data-bbox="354 1904 843 1983">In Servlet, we can override the service() method.</td><td data-bbox="843 1904 1576 1983">In JSP, we cannot override its service() method.</td></tr></tbody></table>	SERVLET	JSP	Servlet is a java code.	JSP is a html based code.	Writing code for servlet is harder than JSP as it is html in java.	JSP is easy to code as it is java in html.	Servlet plays a controller role in MVC approach.	JSP is the view in MVC approach for showing output.	Servlet is faster than JSP.	JSP is slower than Servlet because the first step in JSP lifecycle is the translation of JSP to java code and then compile.	Servlet can accept all protocol requests.	JSP only accept http requests.	In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.
SERVLET	JSP														
Servlet is a java code.	JSP is a html based code.														
Writing code for servlet is harder than JSP as it is html in java.	JSP is easy to code as it is java in html.														
Servlet plays a controller role in MVC approach.	JSP is the view in MVC approach for showing output.														
Servlet is faster than JSP.	JSP is slower than Servlet because the first step in JSP lifecycle is the translation of JSP to java code and then compile.														
Servlet can accept all protocol requests.	JSP only accept http requests.														
In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.														

	<p>In Servlet by default session management is not enabled, user have to enable it explicitly.</p>	<p>In JSP session management is automatically enabled.</p>			
	<p>In Servlet we have to implement everything like business logic and presentation logic in just one servlet file.</p>	<p>In JSP business logic is separated from presentation logic by using javaBeans.</p>			
	<p>Modification in Servlet is a time consuming task because it includes reloading, recompiling and restarting the server.</p>	<p>JSP modification is fast, just need to click the refresh button.</p>			
20.	<p>Discuss the need to use JSTL tags?</p> <h3>III. JSTL (JSP Standard Tag Library)</h3> <p>The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.</p> <p>A. Advantage of JSTL</p> <ol style="list-style-type: none"> Fast Development JSTL provides many tags that simplify the JSP. Code Reusability We can use the JSTL tags on various pages. No need to use scriptlet tag It avoids the use of scriptlet tag. <p>B. JSTL Tags</p> <p>There JSTL mainly provides five types of tags:</p> <table border="1"> <thead> <tr> <th>Tag Name</th><th>Description</th></tr> </thead> </table>			Tag Name	Description
Tag Name	Description				

	<u>Core tags</u>	The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is http://java.sun.com/jsp/jstl/core . The prefix of core tag is c .
	<u>Function tags</u>	The functions tags provide support for string manipulation and string length. The URL for the functions tags is http://java.sun.com/jsp/jstl/functions and prefix is fn .
	<u>Formatting tags</u>	The Formatting tags provide support for message formatting, number date formatting, etc. The URL for the Formatting tags is http://java.sun.com/jsp/jstl/fmt and prefix is fmt .
	<u>XML tags</u>	The XML tags provide flow control, transformation, etc. The URL for the XML tags is http://java.sun.com/jsp/jstl/xml and prefix is x .
	<u>SQL tags</u>	The JSTL SQL tags provide SQL support. The URL for the SQL tags is http://java.sun.com/jsp/jstl/sql and prefix is sql .

PART-B

Q.No	Questions
1.	(i) Integrate how servlets work and its life cycle. (ii) Explain and develop the Servlet API.
2.	(i) Analyze a JavaScript to find factorial of a given number. <pre>function factorial(x) { if (x === 0) { return 1; } return x * factorial(x-1); } console.log(factorial(5));</pre> <p>OUTPUT : 120</p>

(ii) **Differentiate** GET and POST method.

Get vs. Post

There are many differences between the Get and Post request. Let's see these differences:

GET	POST
1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
2) Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.
3) Get request can be bookmarked.	Post request cannot be bookmarked.
4) Get request is idempotent . It means second request will be ignored until response of first request is delivered	Post request is non-idempotent .
5) Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.

GET and POST

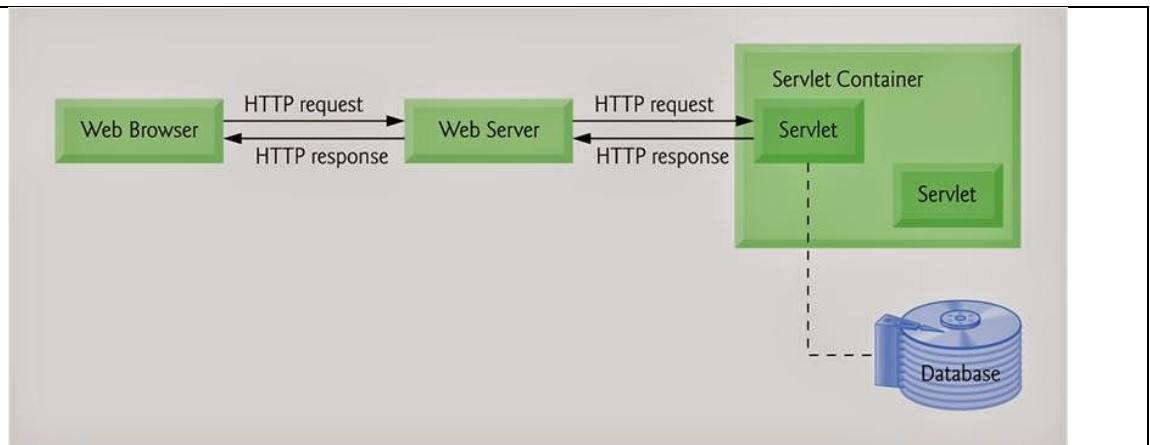
Two common methods for the request-response between a server and client are:

- o **GET**- It requests the data from a specified resource
- o **POST**- It submits the processed data to a specified resource

Demonstrate the Servlet architecture and explain its working principle.

3.

Servlet Architecture



Servlets read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.

Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.

Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.

Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.

Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Servlet API:

Servlet API contains three packages

javax.servlet: Package contains a number of classes and interfaces that describe the contract between a servlet class and the runtime environment provided for an instance of such a class a conforming servelt container.

javax.servlet.annotation: Package contains a number of annotations that allow users to use annotations to declare servlets , filters, listeners and specify the metadata for the declared component

javax.servlet.http: Package contains a number of classes and interfaces that describe and define the contract between a servlet class rnning under the HTTP protocol and the runtime environment provided for an instance of such class by a confirming servlet container.

4.

Consider a database that has a table Employee with two columns Employee Id and Name. Assume that the administrator user id and password to access to access the database table are Scott and Tiger. Write a JDBC program that can query and print all entries in the

	<p>table employee. Make the database using type 2 driver database.driver and connection string jdbc :db.oci.</p>
	<p>Describe in detail the session handling in server side programming.</p> <h4>IV. Managing Session in Servlets</h4> <p>We all know that HTTP is a stateless protocol. All requests and responses are independent. But sometimes you need to keep track of client's activity across multiple requests. For eg. When a User logs into your website, not matter on which web page he visits after logging in, his credentials will be with the server, until he logs out. So this is managed by creating a session.</p> <p>Session Management is a mechanism used by the Web container to store session information for a particular user. There are four different techniques used by Servlet application for session management. They are as follows:</p> <ol style="list-style-type: none"> 1. Cookies 2. Hidden form field 3. URL Rewriting 4. HttpSession <p>Session is used to store everything that we can get from the client from all the requests the client makes.</p>
5.	<p>A. How Session Works</p> <pre> graph LR Client1[Client 1] -- "request id=123" --> WebContainer[Web Container] Client2[Client 2] -- "request id=134" --> WebContainer subgraph WebContainer [Web Container] servlet[servlet] S1[Session id=123] S2[Session id=134] Client1 --- servlet Client2 --- servlet servlet --- S1 servlet --- S2 end </pre> <p>B.</p> <p>The basic concept behind session is, whenever a user starts using our application, we can save a unique identification information about him, in an object which is available throughout the application, until its destroyed. So wherever the user goes, we will always have his information and we can always manage which user is doing what. Whenever a user wants to exit from your application, destroy the object with his information.</p>

	<p>(i) Discuss about JSTL.</p> <h2>V. JSTL (JSP Standard Tag Library)</h2> <p>The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.</p> <h3>A. Advantage of JSTL</h3> <ol style="list-style-type: none"> Fast Development JSTL provides many tags that simplify the JSP. Code Reusability We can use the JSTL tags on various pages. No need to use scriptlet tag It avoids the use of scriptlet tag. <h3>B. JSTL Tags</h3> <p>There JSTL mainly provides five types of tags:</p> <table border="1"> <thead> <tr> <th>Tag Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><u>Core tags</u></td> <td>The JSTL core tag provide variable support, URL management, flow etc. The URL for the core tag is http://java.sun.com/jsp/jstl/core. The prefix of core tag is c.</td> </tr> <tr> <td><u>Function tags</u></td> <td>The functions tags provide support for string manipulation and string conversion. The URL for the functions tags is http://java.sun.com/jsp/jstl/functions and prefix is fn.</td> </tr> <tr> <td><u>Formatting tags</u></td> <td>The Formatting tags provide support for message formatting, number formatting, date formatting, etc. The URL for the Formatting tags is http://java.sun.com/jsp/jstl/fmt and prefix is fmt.</td> </tr> <tr> <td><u>XML tags</u></td> <td>The XML tags provide flow control, transformation, etc. The URL for the XML tags is http://java.sun.com/jsp/jstl/xml and prefix is x.</td> </tr> <tr> <td><u>SQL tags</u></td> <td>The JSTL SQL tags provide SQL support. The URL for the SQL tags is http://java.sun.com/jsp/jstl/sql and prefix is sql.</td> </tr> </tbody> </table> <p>(ii) Summarize a client server JSP program to find simple interest and display the result in client.</p>	Tag Name	Description	<u>Core tags</u>	The JSTL core tag provide variable support, URL management, flow etc. The URL for the core tag is http://java.sun.com/jsp/jstl/core . The prefix of core tag is c .	<u>Function tags</u>	The functions tags provide support for string manipulation and string conversion. The URL for the functions tags is http://java.sun.com/jsp/jstl/functions and prefix is fn .	<u>Formatting tags</u>	The Formatting tags provide support for message formatting, number formatting, date formatting, etc. The URL for the Formatting tags is http://java.sun.com/jsp/jstl/fmt and prefix is fmt .	<u>XML tags</u>	The XML tags provide flow control, transformation, etc. The URL for the XML tags is http://java.sun.com/jsp/jstl/xml and prefix is x .	<u>SQL tags</u>	The JSTL SQL tags provide SQL support. The URL for the SQL tags is http://java.sun.com/jsp/jstl/sql and prefix is sql .
Tag Name	Description												
<u>Core tags</u>	The JSTL core tag provide variable support, URL management, flow etc. The URL for the core tag is http://java.sun.com/jsp/jstl/core . The prefix of core tag is c .												
<u>Function tags</u>	The functions tags provide support for string manipulation and string conversion. The URL for the functions tags is http://java.sun.com/jsp/jstl/functions and prefix is fn .												
<u>Formatting tags</u>	The Formatting tags provide support for message formatting, number formatting, date formatting, etc. The URL for the Formatting tags is http://java.sun.com/jsp/jstl/fmt and prefix is fmt .												
<u>XML tags</u>	The XML tags provide flow control, transformation, etc. The URL for the XML tags is http://java.sun.com/jsp/jstl/xml and prefix is x .												
<u>SQL tags</u>	The JSTL SQL tags provide SQL support. The URL for the SQL tags is http://java.sun.com/jsp/jstl/sql and prefix is sql .												

7.	<p>Explain the use of cookies for tracking for tracking requests with a program.</p> <h2>Session Tracking in JSP</h2> <p>Session Tracking :</p> <p>HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a new connection to the Web server and the server does not keep any record of previous client request.Session tracking is a mechanism that is used to maintain state about a series of requests from the same user(requests originating from the same browser) across some period of time. A session id is a unique token number assigned to a specific user for the duration of that user's session.</p> <p>Need Of Session Tracking :</p> <p>HTTP is a stateless protocol so When there is a series of continuous request and response from a same client to a server, the server cannot identify which client is sending request.If we want to maintain the conversational state,session tracking is needed. For example, in a shopping cart application a client keeps on adding items into his cart using multiple requests.When every request is made,the server should identify in which client's cart the item is to be added. So in this scenario, there is a certain need for session tracking.</p> <p>Solution is, when a client makes a request it should introduce itself by providing unique identifier every time.There are four ways to maintain session between web client and web server.</p> <p>Methods to track session :</p> <ul style="list-style-type: none"> Cookies URL Rewriting Hidden Fields Session API <p>Cookies :</p> <p>Cookies mostly used for session tracking. Cookie is a key value pair of information, sent by the server to the browser. This should be saved by the browser in its space in the client computer.</p>

	<p>Whenever the browser sends a request to that server it sends the cookie along with it. Then the server can identify the client using the cookie.</p> <p>This is not an effective way because many time browser does not support a cookie or users can opt to disable cookies using their browser preferences. In such case, the browser will not save the cookie at client computer and session tracking fails.</p>
8.	<p>(i) Explain about the standard actions in JSP.</p> <p>Actions are used for controlling the behavior of servlet engine.</p> <p>How many standard Action Tags are available in JSP?</p> <p>There are 11 types of Standard Action Tags as following:</p> <ul style="list-style-type: none"> • jsp:useBean • jsp:include • jsp:setProperty • jsp:getProperty • jsp:forward • jsp:plugin • jsp:attribute • jsp:body • jsp:text • jsp:param • jsp:attribute • jsp:output <p>(ii) Analyze MVC architecture of JSP.</p> <h2>VI. MVC in JSP</h2> <ol style="list-style-type: none"> 1. <u>MVC in JSP</u> 2. <u>Example of following MVC in JSP</u> <p>MVC stands for Model View and Controller. It is a design pattern that separates the business logic, presentation logic and data.</p> <p>Controller acts as an interface between View and Model. Controller intercepts all the incoming requests.</p> <p>Model represents the state of the application i.e. data. It can also have business logic.</p>

	<p>View represents the presentation i.e. UI(User Interface).</p> <p>a) Advantage of MVC (Model 2) Architecture</p> <ol style="list-style-type: none"> 1. Navigation Control is centralized 2. Easy to maintain the large application <pre> graph LR View[View (JSP)] --> Model[Model (Java Bean)] Model --> Database[Database] Controller[Controller (Filter/Servlet)] --> View Controller --> Model View --> Controller Database --> Model </pre> <p>The diagram illustrates the MVC (Model 2) architecture. It features a 'Container' boundary defined by a dashed line. Inside this container, there is a 'Controller (Filter/Servlet)' box at the top. Below it are two boxes: 'View (JSP)' on the left and 'Model (Java Bean)' on the right. Arrows point from the View to the Model, and from the Model to the Database. Outside the Container, a monitor icon represents the User Interface (View). An arrow points from the View to the Controller. Another arrow points from the Controller to the Model. A final arrow points from the Database to the Model.</p>
9.	<p>Explain in detail about Servlet database connectivity with an example of student database.</p> <h2>Example of Registration form in servlet</h2> <p>Table creation :</p> <pre> CREATE TABLE "REGISTERUSER" ("NAME" VARCHAR2(4000), "PASS" VARCHAR2(4000), "EMAIL" VARCHAR2(4000), "COUNTRY" VARCHAR2(4000)) /</pre> <h2>Example of Registration form in servlet</h2> <p>In this example, we have created the three pages.</p> <ul style="list-style-type: none"> o register.html o Register.java

- o web.xml

register.html

In this page, we have getting input from the user using text fields and combobox. The information entered by the user is forwarded to Register servlet, which is responsible to store the data into the database.

```
<html>
<body>
<form action="servlet/Register" method="post">

Name:<input type="text" name="userName"/><br/><br/>
Password:<input type="password" name="userPass"/><br/><br/>
Email Id:<input type="text" name="userEmail"/><br/><br/>
Country:
<select name="userCountry">
. <option>India</option>
. <option>Pakistan</option>
. <option>other</option>
. </select>
.
. <br/><br/>
. <input type="submit" value="register"/>
.
. </form>
. </body>
. </html>
```

Register.java

This servlet class receives all the data entered by user and stores it into the database. Here, we are performing the database logic. But you may separate it, which will be better for the web application.

```
import java.io.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class Register extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
}
```

```

. PrintWriter out = response.getWriter();
.

. String n=request.getParameter("userName");
. String p=request.getParameter("userPass");
. String e=request.getParameter("userEmail");
. String c=request.getParameter("userCountry");
.

. try{
.   Class.forName("oracle.jdbc.driver.OracleDriver");
.   Connection con=DriverManager.getConnection(
.     "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
.

.   PreparedStatement ps=con.prepareStatement(
.     "insert into registeruser values(?, ?, ?, ?)");
.

.   ps.setString(1,n);
.   ps.setString(2,p);
.   ps.setString(3,e);
.   ps.setString(4,c);
.

.   int i=ps.executeUpdate();
.   if(i>0)
.     out.print("You are successfully registered...");
.

.

. }catch (Exception e2) {System.out.println(e2);}
.

. out.close();
.
.
.
```

10.	<p>Demonstrate the procedure of installing and configuring Apache Tomcat.</p> <h2>How To Install Apache Tomcat 8 on Ubuntu 16.04</h2> <p>Apache Tomcat is a web server and servlet container that is used to serve Java applications. Tomcat is an open source implementation of the Java Servlet and JavaServer Pages technologies, released by the Apache Software Foundation. This</p>
-----	--

tutorial covers the basic installation and some configuration of the latest release of Tomcat 8 on your Ubuntu 16.04 server.

Step 1: Install Java

Tomcat requires Java to be installed on the server so that any Java web application code can be executed. We can satisfy that requirement by installing OpenJDK with apt-get.

First, update your apt-get package index:

```
sudo apt-get update
```

Then install the Java Development Kit package with apt-get:

```
sudo apt-get install default-jdk
```

Now that Java is installed, we can create a `tomcat` user, which will be used to run the Tomcat service.

Step 2: Create Tomcat User

For security purposes, Tomcat should be run as an unprivileged user (i.e. not root). We will create a new user and group that will run the Tomcat service.

First, create a new `tomcat` group:

```
sudo groupadd tomcat
```

Next, create a new `tomcat` user. We'll make this user a member of the `tomcat` group, with a home directory of `/opt/tomcat` (where we will install Tomcat), and with a shell of `/bin/false` (so nobody can log into the account):

```
sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

Now that our `tomcat` user is set up, let's download and install Tomcat.

Step 3: Install Tomcat

The best way to install Tomcat 8 is to download the latest binary release then configure it manually.

We will install Tomcat to the /opt/tomcat directory. Create the directory, then extract the archive to it with these commands:

```
sudo mkdir /opt/tomcat
```

```
sudo tar xzvf apache-tomcat-8*tar.gz -C /opt/tomcat --strip-components=1
```

Next, we can set up the proper user permissions for our installation.

Step 4: Update Permissions

The tomcat user that we set up needs to have access to the Tomcat installation. We'll set that up now.

Change to the directory where we unpacked the Tomcat installation:

```
cd /opt/tomcat
```

Give the tomcat group ownership over the entire installation directory:

```
sudo chgrp -R tomcat /opt/tomcat
```

Next, give the tomcat group read access to the conf directory and all of its contents, and **execute** access to the directory itself:

```
sudo chmod -R g+r conf
```

```
sudo chmod g+x conf
```

Make the tomcat user the owner of the webapps, work, temp, and logs directories:

```
sudo chown -R tomcat webapps/ work/ temp/ logs/
```

Now that the proper permissions are set up, we can create a systemd service file to manage the Tomcat process.

Step 5: Create a systemd Service File

We want to be able to run Tomcat as a service, so we will set up systemd service file.

Tomcat needs to know where Java is installed. This path is commonly referred to as “JAVA_HOME”. The easiest way to look up that location is by running this command:

Step 6: Adjust the Firewall and Test the Tomcat Server

Now that the Tomcat service is started, we can test to make sure the default page is available.

Before we do that, we need to adjust the firewall to allow our requests to get to the service. If you followed the prerequisites, you will have a ufw firewall enabled currently.

Tomcat uses port 8080 to accept conventional requests. Allow traffic to that port by typing:

```
sudo ufw allow 8080
```

Step 7: Configure Tomcat Web Management Interface

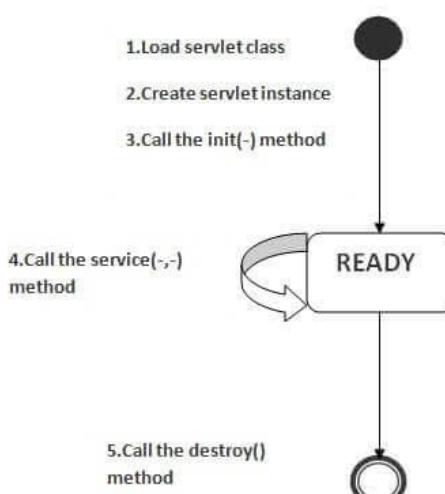
Step 8: Access the Web Interface

Now that we have created a user, we can access the web management interface again in a web browser. Once again, you can get to the correct interface by entering your server’s domain name or IP address followed on port 8080 in your browser:

Open in web browser

`http://server_domain_or_IP:8080`

The page you see should be the same one you were given when you tested earlier:

	<p>Home Documentation Configuration Examples Wiki Mailing Lists Find Help</p> <p>Apache Tomcat/8.0.33</p> <p>If you're seeing this, you've successfully installed Tomcat. Congratulations!</p> <p>The Apache Software Foundation http://www.apache.org/</p> <p> Recommended Reading: Security Considerations HOW-TO Manager Application HOW-TO Clustering/Session Replication HOW-TO</p> <p>Server Status Manager App Host Manager</p> <p>Developer Quick Start</p> <table border="0"> <tr> <td>Tomcat Setup</td><td>Realms & AAA</td><td>Examples</td><td>Servlet Specifications</td></tr> <tr> <td>First Web Application</td><td>JDBC DataSources</td><td></td><td>Tomcat Versions</td></tr> </table> <p>Your installation of Tomcat is complete! You are now free to deploy your own Java web applications!</p>	Tomcat Setup	Realms & AAA	Examples	Servlet Specifications	First Web Application	JDBC DataSources		Tomcat Versions
Tomcat Setup	Realms & AAA	Examples	Servlet Specifications						
First Web Application	JDBC DataSources		Tomcat Versions						
11.	<p>(i) Discuss about the Servlet life cycle.</p> <h2>VII. Life Cycle of a Servlet (Servlet Life Cycle)</h2> <p>The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:</p> <ol style="list-style-type: none"> 1. Servlet class is loaded. 2. Servlet instance is created. 3. init method is invoked. 4. service method is invoked. 5. destroy method is invoked.  <pre> graph TD A(()) -- "1.Load servlet class" --> B[] B -- "2.Create servlet instance" --> C[] C -- "3.Call the init(-) method" --> D[READY] D -- "4.Call the service(-,-) method" --> E((())) E -- "5.Call the destroy() method" --> F((()) </pre> <p>The diagram illustrates the five steps of the servlet life cycle. It starts with a black circle (Step 1), followed by a rounded rectangle (Step 2). Step 3 leads to a rounded rectangle labeled "READY". From the "READY" state, an arrow points to a circle with a diagonal line (Step 4). Finally, an arrow from the circle with a diagonal line points to another circle with a diagonal line (Step 5).</p>								

	<p>As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.</p> <p>(ii) List JSP advantages.</p> <ol style="list-style-type: none"> 1. Advantages of JSP over Servlet <p>There are many advantages of JSP over the Servlet. They are as follows:</p> <ol style="list-style-type: none"> a) 1) Extension to Servlet JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy. b) 2) Easy to maintain JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic. c) 3) Fast Development: No need to recompile and redeploy If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application. d) 4) Less code than Servlet In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.
12.	<p>(i) Explain and write a simple JDBC program.</p> <h3>Java Database Connectivity with MySQL</h3> <p>To connect Java application with the MySQL database, we need to follow 5 following steps.</p> <ol style="list-style-type: none"> 1. Driver class: The driver class for the mysql database is com.mysql.jdbc.Driver. 2. Connection URL: The connection URL for the mysql database is jdbc:mysql://localhost:3306/sonoo where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we

may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.

3. **Username:** The default username for the mysql database is **root**.
4. **Password:** It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

```
create database sonoo;
use sonoo;
create table emp(id int(10),name varchar(40),age int(3));
```

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password both.

```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sonoo","root","root");
//here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
con.close();
} catch(Exception e){ System.out.println(e);}
}
}
```

(ii) **List** various JSP scripting components.

VIII. JSP Scripting Element

JSP Scripting element are written inside `<% %>` tags. These code inside `<% %>` tags are processed by the JSP engine during translation of the JSP page. Any other text in the JSP page is considered as HTML code or plain text.

Example:

```
html>
<head>
```

```

<title>My First JSP Page</title>
</head>
<%
    int count = 0;
%>
<body>
    Page Count is <% out.println(++count); %>
</body>
</html>

```

1. Types of scripting elements

Scripting Element	Example
Comment	<%-- comment --%>
Directive	<%@ directive %>
Declaration	<%! declarations %>
Scriptlet	<% scriptlets %>
Expression	<%= expression %>

B. JSP Comment

JSP Comment is used when you are creating a JSP page and want to put in comments about what you are doing. JSP comments are only seen in the JSP page. These comments are not included in servlet source code during translation phase, nor they appear in the HTTP response. Syntax of JSP comment is as follows :

```
<%-- JSP comment --%>
```

Simple Example of JSP Comment

```

<html>
    <head>
        <title>My First JSP Page</title>
    </head>
    <%
        int count = 0;
    %>
    <body>
        <%-- Code to show page count --%>
        Page Count is <% out.println(++count); %>
    </body>
</html>

```

13.

(i) **Demonstrate** with suitable example for core and formatting tags in JSTL.

JSTL Formatting tags

The formatting tags provide support for message formatting, number and date formatting etc. The url for the formatting tags is <http://java.sun.com/jsp/jstl/fmt> and prefix is **fmt**.

The JSTL formatting tags are used for internationalized web sites to display and format text, the time, the date and numbers. The syntax used for including JSTL formatting library in your JSP is:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
```

Formatting Tags	Descriptions
fmt:parseNumber	It is used to Parses the string representation of a currency, percentage or number.
fmt:timeZone	It specifies a parsing action nested in its body or the time zone for any time formatting.
fmt:formatNumber	It is used to format the numerical value with specific format or precision.
fmt:parseDate	It parses the string representation of a time and date.
fmt:bundle	It is used for creating the ResourceBundle objects which will be used by their tag body.
fmt:setTimeZone	It stores the time zone inside a time zone configuration variable.
fmt:setBundle	It loads the resource bundle and stores it in a bundle configuration variable or the named scoped variable.
fmt:message	It display an internationalized message.
fmt:formatDate	It formats the time and/or date using the supplied pattern and styles.

JSTL Core <c:choose>, <c:when>, <c:otherwise> Tag

The `<c:choose>` tag is a conditional tag that establish a context for mutually exclusive conditional operations. It works like a Java **switch** statement in which we choose between a numbers of alternatives.

Example

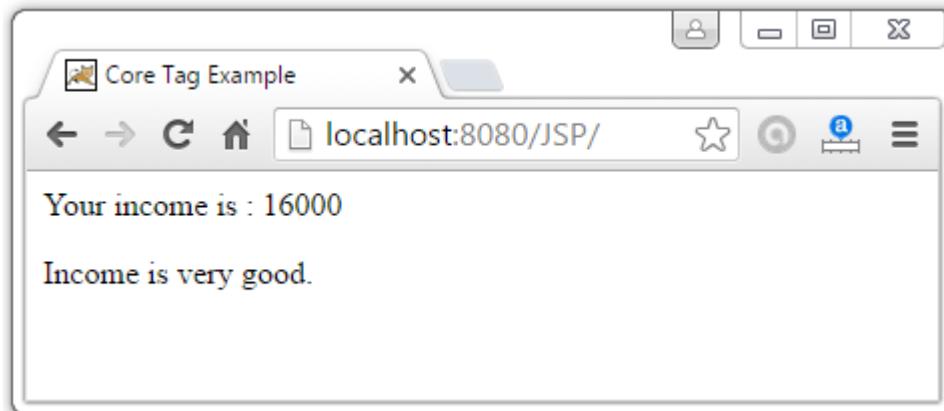
Let's see the simple example of < c:choose >, < c:when > < c:otherwise > tag:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${4000*4}" />
<p>Your income is : <c:out value="${income}" /></p>
<c:choose>
    <c:when test="${income <= 1000}">
        Income is not good.
    </c:when>
    <c:when test="${income > 10000}">
        Income is very good.
    </c:when>
    <c:otherwise>
        Income is undetermined...
    </c:otherwise>
</c:choose>
</body>
</html>
```

This will produce the following result:

Your income is : 16000

Income is very good.



(ii) **Demonstrate** with suitable example for SQL and XML tags in JSTL.

JSTL SQL

The **<sql:setDataSource>** tag sets the data source configuration variable or saves the data-source information in a scoped variable that can be used as input to the other JSTL database actions.

Attribute

The **<sql:setDataSource>** tag has the following attributes –

Attribute	Description
driver	Name of the JDBC driver class to be registered
url	JDBC URL for the database connection
user	Database username
password	Database password
password	Database password
dataSource	Database prepared in advance
var	Name of the variable to represent the database
scope	Scope of the variable to represent the database

Example

Consider the following information about your MySQL database setup –

- We are using **JDBC MySQL** driver.
- We are going to connect to TEST database on local machine.
- We would use **user_id** and **mypassword** to access TEST database.

All the above parameters will vary based on your MySQL or any other database setup. Considering the above parameters, following example uses the **setDataSource** tag –

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>  
<%@ taglib uri = "http://java.sun.com/jsp/jstl/sql" prefix = "sql"%>
```

```

<html>
  <head>
    <title>JSTL sql:setDataSource Tag</title>
  </head>

  <body>
    <sql:setDataSource var = "snapshot" driver = "com.mysql.jdbc.Driver"
      url = "jdbc:mysql://localhost/TEST"
      user = "user_id" password = "mypassword"/>
    <sql:query dataSource = "${snapshot}" sql = "..." var = "result" />
  </body>
</html>

```

JSTL XML

<x:parse> Tag

The <x:parse> tag is used for parse the XML data specified either in the tag body or an attribute. It is used for parse the xml content and the result will stored inside specified variable.

The syntax used for including the <x:parse> tag is:

<x:parse attributes> body content </x:parse>

EXAMPLE

Let us put the following content in **novels.xml** file:

```

<books>
  <book>
    <name>Three mistakes of my life</name>
    <author>Chetan Bhagat</author>
    <price>200</price>
  </book>
  <book>
    <name>Tomorrow land</name>
    <author>NUHA</author>
    <price>2000</price>

```

	<pre></book> </books></pre> <p>index.jsp</p> <p>(IN the same directory)</p> <pre><%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %> <html> <head> <title>x:parse Tag</title> </head> <body> <h2>Books Info:</h2> <c:import var="bookInfo" url="novels.xml"/> <x:parse xml="\${bookInfo}" var="output"/> <p>First Book title: <x:out select="\$output/books/book[1]/name" /></p> <p>First Book price: <x:out select="\$output/books/book[1]/price" /></p> <p>Second Book title: <x:out select="\$output/books/book[2]/name" /></p> <p>Second Book price: <x:out select="\$output/books/book[2]/price" /></p> </body> </html></pre> <p>Output:</p> <p>Books Info:</p> <p>First Book title: Three mistakes of my life</p> <p>First Book price: 200</p> <p>Second Book title: Tomorrow land</p> <p>Second Book price: 2000</p>
14.	<p>Define HTML and JSP. Use the same and design a scientific calculator.</p> <p>Calculator.jsp</p> <pre><%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> <html> <head> <title>Calculator</title></pre>

```
<style>
    h1 {
        font-family: Arial;
        font-size: 14pt;
        font-weight: normal;
    }
    td input {
        font-family: Arial;
        font-size: 10pt;
        width: 30px;
    }

    input.double {
        width: 60px;
    }

    input.doubleheight {
        height: 48px;
    }

    input.display {
        border: 1px solid black;
        readonly: true;
        width: 120px;
        padding: 2px;
    }
</style>
</head>

<body>
    <h1>Calculator</h1>

    <form method="GET" action="calculate.html">
        <table border="0" cellpadding="0" cellspacing="0">
            <tr>
                <td colspan="4">
                    <input class="display" type="text" id="display" value=<c:out
value="${displayAmount}" /> readonly/>
                </td>
            </tr>
            <tr>
                <td><input type="submit" name="button" id="btn-7"
value="7"/></td>
                <td><input type="submit" name="button" id="btn-8"
value="8"/></td>
                <td><input type="submit" name="button" id="btn-9"
value="9"/></td>
                <td><input type="submit" name="button" id="btn-/"
value="/" /></td>
            </tr>
        </table>
    </form>
</body>
```

```

<td><input type="submit" name="button" id="btn-C"
value="C"/></td>
</tr>
<tr>
<td><input type="submit" name="button" id="btn-4"
value="4"/></td>
<td><input type="submit" name="button" id="btn-5"
value="5"/></td>
<td><input type="submit" name="button" id="btn-6"
value="6"/></td>
<td><input type="submit" name="button" id="btn-*"
value="*"/></td>
<td></td>
</tr>
<tr>
<td><input type="submit" name="button" id="btn-1"
value="1"/></td>
<td><input type="submit" name="button" id="btn-2"
value="2"/></td>
<td><input type="submit" name="button" id="btn-3"
value="3"/></td>
<td><input type="submit" name="button" id="btn--" value="-"
"/></td>
<td rowspan="2"><input class="doubleheight" type="submit"
name="button" id="btn-=" value="/" /></td>
</tr>
<tr>
<td colspan="2"><input class="double" type="submit"
name="button" id="btn-0" value="0" /></td>
<td><input type="button" name="button" id="btn-."
value"." /></td>
<td><input type="submit" name="button" id="btn-+"
value "+" /></td>
</tr>
</table>
</form>
</body>
</html>

```

PART – C

Q.No	Questions
1.	Design a HTML forms by embedding JSP code for submission of a resume to a job portal website with appropriate database connectivity.
2.	Evaluate a complete query application for books database using JDBC.
3.	Write a program that allows the user to select a favourite programming language and post the choice to the server. The response is a web page in which the user can click a

	link to view a list of book recommendations. The cookies previously stored on the client are read by the servlet and form a web page containing the book recommendation. Use servlet cookies and HTML.
4.	Develop a JSP program to display the grade of a student by accepting the marks of five subjects.

CS8651 Internet Programming – 2017Reg

PHP: An introduction to PHP – Using PHP – Variables – Program control – Built-in functions – Form Validation – Regular Expressions – File handling – Cookies – Connecting to Database; **XML:** Basic XML – Document Type Definition – XML Schema DOM and Presenting XML, XML Parsers and Validation, XSL and XSLT Transformation, News Feed (RSS and ATOM).

Internet Programming – UNIT-IV

Q.No	Questions
	<p>Define PHP. List the features.</p> <p>What is PHP?</p> <ul style="list-style-type: none">• PHP is an acronym for "PHP: Hypertext Preprocessor"• PHP is a widely-used, open source scripting language• PHP scripts are executed on the server• PHP is free to download and use <p>PHP is an amazing and popular language!</p> <p>It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)! It is deep enough to run the largest social network (Facebook)! It is also easy enough to be a beginner's first server side language!</p> <p>What is a PHP File?</p> <p>1.</p> <ul style="list-style-type: none">• PHP files can contain text, HTML, CSS, JavaScript, and PHP code• PHP code is executed on the server, and the result is returned to the browser as plain HTML• PHP files have extension ".php" <p>What Can PHP Do?</p> <ul style="list-style-type: none">• PHP can generate dynamic page content• PHP can create, open, read, write, delete, and close files on the server• PHP can collect form data• PHP can send and receive cookies• PHP can add, delete, modify data in your database• PHP can be used to control user-access• PHP can encrypt data <p>With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.</p>

List the rules for creating variables in PHP.

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

2.

- A variable starts with the \$ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (\$age and \$AGE are two different variables)
- PHP variable names are case-sensitive

Illustrate a PHP program to determine the type of browser that a web client is using.

Display the Browser – PHP Script

The following PHP function can be used to display the browser:

<?php

```
function get_the_browser()
{
    if(strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== false)
        return 'Internet explorer';
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Trident') !== false)
        return 'Internet explorer';
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Firefox') !== false)
        return 'Mozilla Firefox';
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Chrome') !== false)
        return 'Google Chrome';
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Opera Mini') !== false)
        return "Opera Mini";
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Opera') !== false)
        return "Opera";
    elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Safari') !== false)
        return "Safari";
    else
        return 'Other';
```

3.

```
}
```

```
?>
```

In the above code, we are checking each possible browser that may be and return the browser name. Here we haven't checked the Mozilla because of most of the browser using this as the user agent string.

Below is how to display the browser name on our web page:

```
Echo get_the_browser();
```

Name any four built-in functions in PHP.

PHP Reference

The PHP reference contains different categories of all PHP functions and constants, along with examples.



Infer when should the super global arrays in PHP be used?

Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes.

PHP Global Variables - Superglobals

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- `$GLOBALS`
- `$_SERVER`
- `$_REQUEST`
- `$_POST`
- `$_GET`
- `$_FILES`
- `$_ENV`
- `$_COOKIE`
- `$_SESSION`

Which super global array in PHP would contain a HTML form's POST data?

PHP Superglobal - `$_POST`

Super global variables are built-in variables that are always available in all scopes.

PHP \$ POST

PHP `$_POST` is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". `$_POST` is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the `<form>` tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable `$_POST` to collect the value of the input field:

Example

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
    Name: <input type="text" name="fname">
    <input type="submit">
</form>
```

```

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>

```

Classify the difference between echo() & print() functions.

PHP echo and print Statements

With PHP, there are two basic ways to get output: echo and print.

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small:

The PHP echo Statement	The PHP print Statement
<ul style="list-style-type: none"> ➤ echo has no return value ➤ echo can take multiple parameters ➤ echo is marginally faster than print. 	<ul style="list-style-type: none"> ➤ print has a return value of 1 so it can be used in expressions. ➤ print can take one argument
The echo statement can be used with or without parentheses: echo or echo().	The print statement can be used with or without parentheses: print or print().
Example-1 <?php echo "<h2>PHP is Fun!</h2>"; echo "Hello world! "; echo "I'm about to learn PHP! "; echo "This ", "string ", "was ", "made ", "with multiple parameters. "; ?>	Example-1 <?php print "<h2>PHP is Fun!</h2>"; print "Hello world! "; print "I'm about to learn PHP!"; ?>
PHP is Fun! Hello world!	PHP is Fun! Hello world!

	<p>Example-2</p> <pre><?php \$txt1 = "Learn PHP"; \$txt2 = "W3Schools.com"; \$x = 5; \$y = 4; echo "<h2>" . \$txt1 . "</h2>"; echo "Study PHP at " . \$txt2 . "
"; echo \$x + \$y; ?></pre>	<p>Example-2</p> <pre><?php \$txt1 = "Learn PHP"; \$txt2 = "W3Schools.com"; \$x = 5; \$y = 4; print "<h2>" . \$txt1 . "</h2>"; print "Study PHP at " . \$txt2 . "
"; print \$x + \$y; ?></pre>									
	<p>Learn PHP Study PHP at W3Schools.com 9</p>	<p>Learn PHP Study PHP at W3Schools.com 9</p>									
7.	<p>List any two advantages of XML document.</p> <p>Using XML to exchange information offers many benefits.</p> <p>Advantages of XML include the following:</p> <ul style="list-style-type: none"> ➤ XML uses human, not computer, language. XML is readable and understandable, even by novices, and no more difficult to code than HTML. ➤ XML is completely compatible with Java™ and 100% portable. Any application that can process XML can use your information, regardless of platform. ➤ XML is extendable. Create your own tags, or use tags created by others, that use the natural language of your domain, that have the attributes you need, and that makes sense to you and your users. 										
8.	<p>Give the difference between DTD and XML schema for defining XML document structure with appropriate examples.</p> <h3>DTD vs XSD</h3> <p>There are many differences between DTD (Document Type Definition) and XSD (XML Schema Definition). In short, DTD provides less control on XML structure whereas XSD (XML schema) provides more control.</p> <p>The important differences are given below:</p> <table border="1"> <thead> <tr> <th>No.</th><th>DTD(Document Type Definition)</th><th>XSD(XML Schema Definition)</th></tr> </thead> <tbody> <tr> <td>1)</td><td>DTD stands for Document Type Definition.</td><td>XSD stands for XML Schema Definition.</td></tr> <tr> <td>2)</td><td>DTDs are derived from SGML syntax.</td><td>XSDs are written in XML.</td></tr> </tbody> </table>		No.	DTD(Document Type Definition)	XSD(XML Schema Definition)	1)	DTD stands for Document Type Definition.	XSD stands for XML Schema Definition.	2)	DTDs are derived from SGML syntax.	XSDs are written in XML.
No.	DTD(Document Type Definition)	XSD(XML Schema Definition)									
1)	DTD stands for Document Type Definition.	XSD stands for XML Schema Definition.									
2)	DTDs are derived from SGML syntax.	XSDs are written in XML.									

	3)	DTD doesn't support datatypes.	XSD supports datatypes for elements and attributes.	
	4)	DTD doesn't support namespace.	XSD supports namespace.	
	5)	DTD doesn't define order for child elements.	XSD defines order for child elements.	
	6)	DTD is not extensible.	XSD is extensible.	
	7)	DTD is not simple to learn.	XSD is simple to learn because you don't need to learn new language.	
	8)	DTD provides less control on XML structure.	XSD provides more control on XML structure.	
	Analyze about Query String in PHP.			
9.	<h2>Query string</h2> <p>The information can be sent across the web pages. This information is called query string. This query string can be passed from one page to another by appending it to the address of the page. You can pass more than one query string by inserting the & sign between the query strings. A query string can contain two things: the query string ID and its value. The query string passed across the web pages is stored in <code>\$_REQUEST</code>, <code>\$_GET</code>, or <code>\$_POST</code> variable.</p> <hr/> <p>Query string handling in PHP</p> <hr/> <p>Query strings</p> <hr/> <p>To access the data in a query string you can use the <code>\$_GET</code> global array. Each element in this array has a key which is the name of the query string variable and a value which is the value of that variable.</p> <hr/> <pre>my link</pre>			

This link loads the page mypage.php with two variables *variable1* and *variable2* with values value1 and value2 respectively.

```
echo $_GET['variable1'];
echo $_GET['variable2'];
// outputs:
//value1
//value2
```

Form data

The get method of forms sends the data to a page via a query string.

```
<form name="form1" id="form1" method="get" action="">
    <input name="textbox" id="textbox" type="text" value="value1" />
    <input name="textbox2" id="textbox2" type="text" value="value2" />
    <input type="submit" name="submitbutton" id="submitbutton" value="Submit" />
</form>
```

This form passes the value of the two text boxes to the page myform.php.

```
print_r($_GET);
// outputs:
// Array (
//   [textbox] => value1
//   [textbox2] => value2
//   [submitbutton] => Submit
// )

echo $_GET['textbox'];
//outputs: value1
```

Show an example for XML namespace.

10. A **Namespace** is a set of unique names. Namespace is a mechanisms by which element and attribute name can be assigned to a group. The Namespace is identified by URI(Uniform Resource Identifiers).

Namespace Declaration

A Namespace is declared using reserved attributes. Such an attribute name must either be **xmlns** or begin with **xmlns:** shown as below –

```
<element xmlns:name = "URL">
```

Syntax

- The Namespace starts with the keyword **xmlns**.
- The word **name** is the Namespace prefix.
- The **URL** is the Namespace identifier.

Example

Namespace affects only a limited area in the document. An element containing the declaration and all of its descendants are in the scope of the Namespace. Following is a simple example of XML Namespace –

```
<?xml version = "1.0" encoding = "UTF-8"?>
<cont:contact xmlns:cont = "www.tutorialspoint.com/profile">
    <cont:name>Tanmay Patil</cont:name>
    <cont:company>Tutorialspoint</cont:company>
    <cont:phone>(011) 123-4567</cont:phone>
</cont:contact>
```

Here, the Namespace prefix is **cont**, and the Namespace identifier (URI) as www.tutorialspoint.com/profile. This means, the element names and attribute names with the **cont** prefix (including the contact element), all belong to the www.tutorialspoint.com/profile namespace.

Define XML parse tree.

An XML document is always descriptive. The tree structure is often referred to as **XML Tree** and plays an important role to describe any XML document easily.

The tree structure contains root (parent) elements, child elements and so on. By using tree structure, you can get to know all succeeding branches and sub-branches starting from the root. The parsing starts at the root, then moves down the first branch to an element, take the first branch from there, and so on to the leaf nodes.

11.

Example

Following example demonstrates simple XML tree structure –

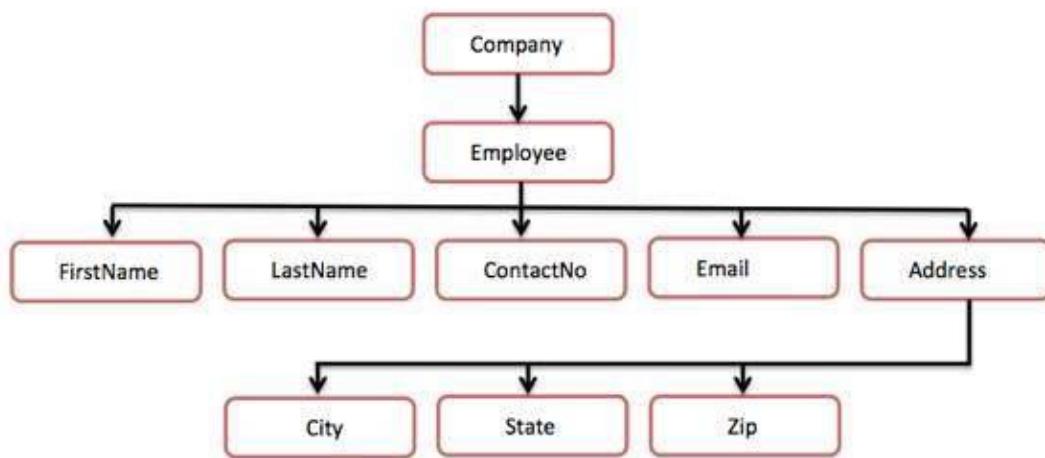
```
<?xml version = "1.0"?>
<Company>
    <Employee>
        <FirstName>Tanmay</FirstName>
```

```

<LastName>Patil</LastName>
<ContactNo>1234567890</ContactNo>
<Email>tanmaypatil@xyz.com</Email>
<Address>
    <City>Bangalore</City>
    <State>Karnataka</State>
    <Zip>560212</Zip>
</Address>
</Employee>
</Company>

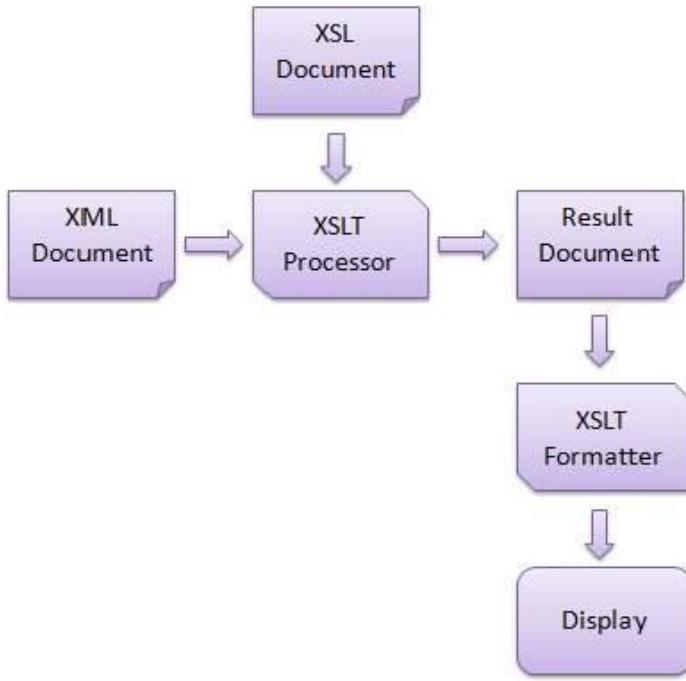
```

Following tree structure represents the above XML document –



In the above diagram, there is a root element named as <company>. Inside that, there is one more element <Employee>. Inside the employee element, there are five branches named <FirstName>, <LastName>, <ContactNo>, <Email>, and <Address>. Inside the <Address> element, there are three sub-branches, named <City> <State> and <Zip>.

12.	<p>Identify why XSLT is an important tool for development of web applications.</p> <h2>What is XSLT</h2> <p>XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.</p> <h3>How XSLT Works</h3> <p>An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.</p>
-----	---



Advantages

Here are the advantages of using XSLT –

- Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
- Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

Assess the data types in XML schema.

You can define XML schema elements in the following ways –

Simple Type

Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example –

```
<xs:element name = "phone_number" type = "xs:int" />
```

Complex Type

13. A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example –

```
<xs:element name = "Address">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string" />
      <xs:element name = "company" type = "xs:string" />
      <xs:element name = "phone" type = "xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

	<pre></xs:complexType> </xs:element></pre>
14.	<p>Explain DTD for XML Schemas.</p> <p>A document type definition (DTD) is a set of <i>markup declarations</i> that define a <i>document type</i> for a SGML-family markup language (GML, SGML, XML, HTML).</p> <p>A DTD defines the valid building blocks of an XML document. It defines the document structure with a list of validated elements and attributes. A DTD can be declared inline inside an XML document, or as an external reference.</p> <h3><u>XML DTD schema example</u></h3> <p>An example of a very simple external XML DTD to describe the schema of a list of persons might consist of:</p> <pre><!ELEMENT people_list (person)*> <!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)> <!ELEMENT name (#PCDATA)> <!ELEMENT birthdate (#PCDATA)> <!ELEMENT gender (#PCDATA)> <!ELEMENT socialsecuritynumber (#PCDATA)></pre> <p>Taking this line by line:</p> <ol style="list-style-type: none"> 1. <code>people_list</code> is a valid element name, and an instance of such an element contains any number of <code>person</code> elements. The <code>*</code> denotes there can be 0 or more <code>person</code> elements within the <code>people_list</code> element. 2. <code>person</code> is a valid element name, and an instance of such an element contains one element named <code>name</code>, followed by one named <code>birthdate</code> (optional), then <code>gender</code> (also optional) and <code>socialsecuritynumber</code> (also optional). The <code>?</code> indicates that an element is optional. The reference to the <code>name</code> element name has no <code>?</code>, so a <code>person</code> element <i>must</i> contain a <code>name</code> element. 3. <code>name</code> is a valid element name, and an instance of such an element contains "parsed character data" (<code>#PCDATA</code>). 4. <code>birthdate</code> is a valid element name, and an instance of such an element contains parsed character data. 5. <code>gender</code> is a valid element name, and an instance of such an element contains parsed character data. 6. <code>socialsecuritynumber</code> is a valid element name, and an instance of such an element contains parsed character data. <p>An example of an XML file that uses and conforms to this DTD follows. The DTD is referenced here as an external subset, via the SYSTEM specifier and a URI. It assumes that we can identify the DTD with the relative URI reference "example.dtd"; the</p>

"people_list" after "!DOCTYPE" tells us that the root tags, or the first element defined in the DTD, is called "people_list":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

Evaluate the process of displaying XML document in browser.

Display an XML Document in a Web Browser

Displaying XML Using CSS

XML stands for Extensible Markup Language. It is a dynamic markup language. It is used to transform data from one form to another form.

An XML file can be displayed using two ways. These are as follows :-

1. Cascading Style Sheet
2. Extensible Stylesheet Language Transformation

Displaying XML file using CSS :

CSS can be used to display the contents of the XML document in a clear and precise manner. It gives the design and style to whole XML document.

- **Basic steps in defining a CSS style sheet for XML :**

For defining the style rules for the XML document, the following things should be done :-

1. Define the style rules for the text elements such as font-size, color, font-weight, etc.
2. Define each element either as a block, inline or list element, using the display property of CSS.
3. Identify the titles and bold them.

- **Linking XML with CSS :**

In order to display the XML file using CSS, link XML file with CSS. Below is the syntax for linking the XML file with CSS:

```
<?xml-stylesheet type="text/css" href="name_of_css_file.css"?>
```

- **Example 1.**

In this example, the XML file is created that contains the information about five books and displaying the XML file using CSS.

XML file :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="Rule.css"?>
<books>
```

```

<heading>Welcome To GeeksforGeeks </heading>
<book>
    <title>Title :- Web Programming</title>
    <author>Author :- Chrisbates</author>
    <publisher>Publisher :- Wiley</publisher>
    <edition>Edition :- 3</edition>
    <price> Price :- 300</price>
</book>
<book>
    <title>Title :- Internet world-wide-web</title>
    <author>Author :- Ditel</author>
    <publisher>Publisher :- Pearson</publisher>
    <edition>Edition :- 3</edition>
    <price>Price :- 400</price>
</book>

</books>

```

In the above example, Books.xml is linked with Rule.css which contains the corresponding style sheet rules.

CSS FILE :

```

books {
    color: white;
    background-color : gray;
    width: 100%;
}
heading {
    color: green;
    font-size : 40px;
    background-color : powderblue;
}
heading, title, author, publisher, edition, price {
    display : block;
}
title {
    font-size : 25px;
    font-weight : bold;
}

```

- **Output :**



XML namespaces are used for providing uniquely named [elements](#) and attributes in an [XML](#) document. They are defined in a [W3C recommendation](#).^{[1][2]} An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a [namespace](#), the ambiguity between identically named elements or attributes can be resolved.

How to get rid of name conflict?

1) By Using a Prefix

You can easily avoid the XML namespace by using a name prefix.

```
<h:table>
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Note: In this example, you will get no conflict because both the tables have specific names.

2) By Using xmlns Attribute

You can use xmlns attribute to define namespace with the following syntax:

```
<element xmlns:name = "URL">
```

Let's see the example:

```
<root>
<h:table xmlns:h="http://www.abc.com/TR/html4/">
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.xyz.com/furniture">
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
```

```
. <f:length>120</f:length>
. </f:table>
. </root>
```

In the above example, the `<table>` element defines a namespace and when a namespace is defined for an element, the child elements with the same prefixes are associated with the same namespace.

Analyze on ATOM in RSS.

What is Atom 1.0 ?

Atom is the name of an XML-based Web content and metadata syndication format, and an application-level protocol for publishing and editing Web resources belonging to periodically updated websites.

Atom is a relatively recent spec and is much more robust and feature-rich than RSS. For instance, where RSS requires descriptive fields such as title and link only in item breakdowns, Atom requires these things for both items and the full Feed.

All Atom Feeds must be well-formed [XML](#) documents, and are identified with the `application/atom+xml` media type.

Structure of an Atom 1.0 Feed

A Feed consists of some metadata, followed by any number of entries. Here is a basic structure of an Atom 1.0 Feed.

17.

```
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <title>...</title>
    <link>...</link>
    <updated>...</updated>

    <author>
        <name>...</name>
    </author>

    <id>...</id>

    <entry>
        <title>...</title>
        <link>...</link>
        <id>...</id>

        <updated>...</updated>
        <summary>...</summary>
    </entry>

</feed>
```

	<p>Atom 1.0 Feed Tags</p> <p>An Atom 1.0 Feed Document will be constructed of the following two elements:</p> <ul style="list-style-type: none"> • <u><feed></u> Elements • <u><entry></u> Elements
18.	<p>Summarize the advantage of RSS documents?</p> <p style="text-align: center;">RSS - Advantages</p> <p>RSS is taking off so quickly because people are liking it. RSS is easy to use and it has advantages for a publisher as well as for a subscriber. Here we have listed out a few advantages of RSS for subscribers as well as for publishers.</p> <h3>Advantages for Subscribers</h3> <p>RSS subscribers are the people who subscribe to read a published Feed. Here are some of the advantages of RSS Feeds for subscribers:</p> <ul style="list-style-type: none"> • All news at one place: You can subscribe to multiple news groups and then you can customize your reader to have all the news on a single page. It will save you a lot of time. • News when you want it: Rather than waiting for an e-mail, you go to your RSS reader when you want to read a news. Furthermore, RSS Feeds display more quickly than information on web-sites, and you can read them offline if you prefer. • Get the news you want: RSS Feed comes in the form of headlines and a brief description so that you can easily scan the headlines and click only those stories that interest you. • Freedom from e-mail overload: You are not going to get any email for any news or blog update. You just go to your reader and you will find updated news or blog automatically whenever there is a change on the RSS server. • Easy republishing: You may be both a subscriber and a publisher. For example, you may have a web-site that collects news from various other sites and then republishes it. RSS allows you to easily capture that news and display it on your site. <h3>Advantages for Publishers</h3> <p>RSS publishers are the people who publish their content through RSS feed. We would suggest you to use RSS:</p> <ul style="list-style-type: none"> • if you want to get your message out and easily, • if you want people to see what you publish, and • if you want your news to bring people back to your site. <p>Here are some of the advantages of RSS if you publish on the Web:</p>

	<ul style="list-style-type: none"> • Easier publishing: RSS is really simple publishing. You don't have to maintain a database of subscribers to send your information to them, instead they will access your Feed using a reader and will get updated content automatically. • A simpler writing process: If you have a new content on your web site, you only need to write an RSS Feed in the form of titles and short descriptions, and link back to your site. • An improved relationship with your subscribers: Because people subscribe from their side, they don't feel as if you are pushing your content on them. • The assurance of reaching your subscribers: RSS is not subject to spam filters, your subscribers get the Feeds, which they subscribe to and nothing more. • Links back to your site: RSS Feeds always include links back to a website. It directs a lot of traffic towards your website. • Relevance and timeliness: Your subscribers always have the latest information from your site.
19.	<p>Rewrite the declaration for elements in XML.</p>
	<p>How would you prepare the steps to get the RSS file on web?</p> <h3>Uploading an RSS Feed</h3> <p>Here are the simple steps to put your RSS Feed on the web.</p> <ul style="list-style-type: none"> • First decide which version of RSS Feed you are going to use for your site. We would recommend you to use the latest version available. • Create your RSS Feed in a text file with extension either .xml or .rdf. Upload this file on your web server. • You should validate your RSS Feed before making it live. Check the next chapter on RSS Feed Validation. • Create a link on your Web Pages for the RSS Feed file. You will use a small yellow button for the link that says either RSS or XML.
20.	<p>Now, your RSS Feed is online and people can start using it. But there are ways to promote your RSS Feed so that more number of people can use your RSS Feed.</p> <h3>Promote Your RSS Feed</h3> <ul style="list-style-type: none"> • Submit your RSS Feed to the RSS Feed Directories. There are many directories available on the web, where you can register your Feed. Some of them are given here: <ul style="list-style-type: none"> ○ <u>Syndic8</u>: Over 300,000 Feeds listed. ○ <u>Daypop</u>: Over 50,000 feeds listed. ○ <u>Newsisfree</u>: Over 18,000 Feeds. • Register your Feed with the major search engines. Similar to your web pages, you can add your Feed as well with the following major search engines. <ul style="list-style-type: none"> ○ Yahoo - http://publisher.yahoo.com/promote.php

- Google - <http://www.google.com/webmasters/add.html>
- Bing - <http://www.bing.com/toolbox/submit-site-url>

Keeping Up-To-Date Feed

As we have explained earlier, RSS Feed makes sense for the site which are changing their content very frequently, for example, any news or blogging sites.

So now, you have got RSS Feed buttons from Google, Yahoo, and MSN. You must make sure to update your content frequently and that your RSS Feed is constantly available.

PART-B

- (i) **Describe** about the introduction and installation of PHP.

Introduction to PHP

PHP is one of the most widely used server side scripting language for web development. Popular websites like Facebook, Yahoo, Wikipedia etc are developed using PHP.

PHP is so popular because it's very simple to learn, code and deploy on server, hence it has been the first choice for beginners since decades.

Uses of PHP

To further fortify your trust in PHP, here are a few applications of this amazing scripting language:

1. It can be used to **create Web applications** like Social Networks(Facebook, Digg), Blogs(Wordpress, Joomla), eCommerce websites(OpenCart, Magento etc.) etc.
2. **Common Line Scripting.** You can write PHP scripts to perform different operations on any machine, all you need is a PHP parser for this.
3. **Create Facebook applications** and easily integrate Facebook plugins in your website, using Facebook's PHP SDK. Check this [link](#) for more information.
4. **Sending Emails** or building email applications because PHP provides with a robust email sending function.
5. Wordpress is one of the most used blogging(CMS) platform in the World, and if you know PHP, you can try a hand in **Wordpress plugin development**.

Manual Installation

Step 1: Download the files. Download the latest **PHP 5 ZIP** package from www.php.net/downloads.php. ...

Step 2: Extract the files. ...

Step 3: Configure **php**. ...

Step 4: Add C:\php to the path environment variable. ...

Step 5: Configure **PHP** as an Apache module. ...

Step 6: Test a **PHP** file.

(ii) **Design** simple calculator using PHP.

Calculator.php

<!DOCTYPE html>

```
<html>
    <head>
        <title>Simple Calculator In PHP | Webdevtrick.com</title>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <link
            href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css"
            rel="stylesheet">
    </head>
    <body>

        <div class="container" style="margin-top: 50px">

            <?php

                // If the submit button has been pressed
                if(isset($_POST['submit']))
                {
                    // Check number values
                    if(is_numeric($_POST['number1']) && is_numeric($_POST['number2']))
                    {
                        // Calculate total
                        if($_POST['operation'] == 'plus')
                        {
                            $total = $_POST['number1'] + $_POST['number2'];
                        }
                        if($_POST['operation'] == 'minus')
                        {
                            $total = $_POST['number1'] - $_POST['number2'];
                        }
                        if($_POST['operation'] == 'multiply')
                        {
                            $total = $_POST['number1'] * $_POST['number2'];
                        }
                    }
                }
            </?php>

        </div>
    </body>
</html>
```

```

        }
        if($_POST['operation'] == 'divided by')
        {
            $total = $_POST['number1'] / $_POST['number2'];
        }

        // Print total to the browser
        echo "<h1>{$_POST['number1']} {$_POST['operation']}
```

{\$_POST['number2']} equals {\$total}</h1>";

```

    } else {

        // Print error message to the browser
        echo 'Numeric values are required';

    }
}
// end PHP. Code by webdevtrick.com
?>

<!-- Calculator form by webdevtrick.com -->
<form method="post" action="calculator.php">
    <input name="number1" type="text" class="form-control" style="width: 150px; display: inline" />
    <select name="operation">
        <option value="plus">Plus</option>
        <option value="minus">Minus</option>
        <option value="multiply">Multiply</option>
        <option value="divided by">Divide</option>
    </select>
    <input name="number2" type="text" class="form-control" style="width: 150px; display: inline" />
    <input name="submit" type="submit" value="Calculate" class="btn btn-primary" />
</form>

</div>

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

Explain about control statements and data types in PHP with example.

Control Statements in PHP with Examples

2.

Like any other languages, PHP is built out of a series of control statements. The control statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing or an empty statement.

In PHP we have the following conditional statements:

if statement – We use this control statement to execute some code only if a specified condition is true.

if...else statement – We use this control statement to execute some code if a condition is true and another code if the condition is false.

if...elseif....else statement – We use this control statement to select one of several blocks of code to be executed

switch statement – We use this control statement to select one of many blocks of code to be executed

1. The if Statement

Use the if statement to execute some code only if a specified condition is true. The expression is evaluated to its Boolean value. If expression evaluates to TRUE, PHP will execute statement, and if it evaluates to FALSE – it'll ignore it

Syntax

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

The following example would display " A is bigger than B" if \$a is bigger than \$b:

```
<?php  
if ($a > $b)  
echo "A is bigger than B";  
?>
```

2. The if...else Statement

elseif, as its name suggests, is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if expression evaluates to FALSE. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to TRUE.

```
if (condition)  
    code to be executed if condition is true;  
else  
    code to be executed if condition is false;
```

For example, the following code would display a is bigger than b, a equal to b or a is smaller than b:

```
<?php  
if ($a > $b) {  
    echo "a is bigger than b";  
} elseif ($a == $b) {  
    echo "a is equal to b";  
} else {  
    echo "a is smaller than b";  
}  
?>
```

3. The if...elseif....else Statement

4. The Switch Statement

The switch statement is similar to IF statements on the same expression. In many occasions,

	<p>Use the if....elseif...else statement to select one of several blocks of code to be executed.</p> <pre>if (condition) code to be executed if condition is true; elseif (condition) code to be executed if condition is true; else code to be executed if condition is false;</pre> <p>Note: Note that elseif and else if will only be considered exactly the same when using curly brackets as in the above example. When using a colon to define your if/elseif conditions, you must not separate else if into two words, or PHP will fail with a parse error.</p>	<p>you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the switch statement is for.</p> <pre>switch () { case condition1 break; case condition2 break; }</pre> <p>For example, the following code would display \$i matched value as 0 or 1 or 2:</p> <pre><?php switch (\$i) { case 0: echo "i equals 0"; case 1: echo "i equals 1"; case 2: echo "i equals 2"; } ?></pre>
--	--	---

3.	<p>(i) Create an XML document that marks up various sports and their descriptions. Use XSLT to tabulate neatly the elements and attributes of the document.</p> <pre><?xml version="1.0"?> -<events league="WC Falun" tournament="2010/2011" template="World Cup" sport="Cross Country Skiing" ut="2012-09-05" id="821135"></pre> <p>-<event id="866683" status="Finished" round="8001 - 1/1 (Final)" date="2011-03-20 13:15:00" name="10 km Freestyle Handicap Pursuit"></p> <p>-<results participantname="Marit Bjoergen" participantid="43427"></p> <p><result id="9498426" value="1" type="rank"/></p> <p><result id="9498424" value="27:58.0" type="duration"/></p>
----	--

```

<result id="9505038" value="200" type="points"/>

<result id="9498425" value="" type="comment"/>

<result id="9497448" value="1" type="startnumber"/>

</results>

-<results participantname="Justyna Kowalczyk" participantid="43775">

<result id="9498429" value="2" type="rank"/>

<result id="9498427" value="+1:58.0" type="duration"/>

<result id="9505039" value="160" type="points"/>

<result id="9498428" value="" type="comment"/>

<result id="9497454" value="2" type="startnumber"/>

</results>
</event></events>
```

(ii) **Illustrate** a JSP page that enables the user to input the first name and in response outputs the last name.

POST Method Example Using Form

Below is the **main.jsp** JSP program to handle the input given by web browser using the GET or the POST methods.

```

<html>
  <head>
    <title>Using GET and POST Method to Read Form Data</title>
  </head>

  <body>
    <center>
      <h1>Using POST Method to Read Form Data</h1>

      <ul>
        <li><p><b>First Name:</b>
          <%= request.getParameter("first_name")%>
        </p></li>
        <li><p><b>Last Name:</b>
          <%= request.getParameter("last_name")%>
        </p></li>
      </ul>
```

```
</body>  
</html>
```

Following is the content of the **Hello.htm** file –

```
<html>  
  <body>  
  
    <form action = "main.jsp" method = "POST">  
      First Name: <input type = "text" name = "first_name">  
      <br />  
      Last Name: <input type = "text" name = "last_name" />  
      <input type = "submit" value = "Submit" />  
    </form>  
  
  </body>  
</html>
```

Let us now keep **main.jsp** and **hello.htm** in **<Tomcat-installationdirectory>/webapps/ROOT directory**. When you access **http://localhost:8080/Hello.htm**, you will receive the following output.

First Name:
Last Name:

Try to enter the First and the Last Name and then click the submit button to see the result on your local machine where tomcat is running.

4. **Create** a webserver based chat application using PHP. The application should provide the following functions Login, Send message (to one or more contacts) and Receive messages (from one or more contacts)

5. (i) **Write** a PHP program that tests whether an email address is input correctly. Test your program with both valid and invalid email addresses.

PHP - Validate Name, E-mail, and URL

Example

```
<?php  
// define variables and set to empty values
```

```

$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
        // check if name only contains letters and whitespace
        if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
            $nameErr = "Only letters and white space allowed";
        }
    }

    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = test_input($_POST["email"]);
        // check if e-mail address is well-formed
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = "Invalid email format";
        }
    }

    if (empty($_POST["website"])) {
        $website = "";
    } else {
        $website = test_input($_POST["website"]);
        // check if URL address syntax is valid (this regular expression
        // also allows dashes in the URL)
        if (!preg_match("/\b(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\%/?=~_|!:,.;]*[-a-z0-9+&@#\%=~_|]/i",$website)) {
            $websiteErr = "Invalid URL";
        }
    }

    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = test_input($_POST["comment"]);
    }

    if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
    } else {
        $gender = test_input($_POST["gender"]);
    }
}

```

```
}
```

```
?>
```

OUTPUT

PHP Form Validation Example

* required field

Name: *

E-mail: *

Website:

Comment:

Gender: Female Male Other *

Your Input:

Anand M
anand@ibm.com
www.ibm.com
Yhis is a comment
male

Identify and explain about database connectivity illustrate PHP connectivity with any of the databases.

6.

METHOD FOR: CONNECTING TO MYSQL USING MYSQL

The MySQL Improved extension uses the *mysqli* class, which replaces the set of legacy MySQL functions.

To connect to MySQL using the MySQL Improved extension, follow these steps:

1. Use the following PHP code to connect to MySQL and select a database. Replace *username* with your username, *password* with your password, and *dbname* with the database name:

```
2.    <?php  
3.        $mysqli = new mysqli("localhost", "username", "password", "dbname");  
    ?>
```

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = mysqli_connect($servername, $username, $password);  
  
// Check connection  
if (!$conn) {  
    die("Connection failed: " . mysqli_connect_error());  
}  
echo "Connected successfully";  
?>
```

- (i) Discuss on methods for using cookies in PHP.

Cookies in PHP

Cookies are used to store the information of a web page in a remote browser, so that when the same user comes back to that page, that information can be retrieved from the browser itself.

7. **Uses of cookie**

Cookies are often used to perform following tasks:

- **Session management:** Cookies are widely used to manage user sessions. For example, when you use an online shopping cart, you keep adding items in the cart and finally when you checkout, all of those items are added to the list of items you have purchased. This can be achieved using cookies.
- **User identification:** Once a user visits a webpage, using cookies, that user can be remembered. And later on, depending upon the search/visit pattern of the user,

content which the user likely to be visited are served. A good example of this is 'Retargetting'. A concept used in online marketing, where depending upon the user's choice of content, advertisements of the relevant product, which the user may buy, are served.

- **Tracking / Analytics:** Cookies are used to track the user. Which, in turn, is used to analyze and serve various kind of data of great value, like location, technologies (e.g. browser, OS) form where the user visited, how long (s)he stayed on various pages etc.

How to create a cookie in PHP

PHP has a setcookie() function to send a cookie. We will discuss this function in detail now.

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

setcookie() returns boolean.

Example:

Following example shows how to create a cookie in PHP.

```
<?php  
  
$cookie_value = "w3resource tutorials";  
  
setcookie("w3resource", $cookie_value, time()+3600, "/home/your_username/",  
"example.com", 1, 1);  
  
if (isset($_COOKIE['cookie']))  
  
echo $_COOKIE["w3resource"];  
  
?>
```

(ii) **Give a note on regular expressions.**

Summarize in detail the XML schema, built in and user defined data types.

What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

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

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

XSD Simple Elements

XML Schemas define the elements of your XML files.

A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.

The text can be of many different types like boolean, string, date, etc.), or it can be a custom type that you can define yourself.

You can also add restrictions (facets) to a data type in order to limit its content, or you can require the data to match a specific pattern.

The syntax for defining a simple element is:

```
<xs:element name="xxx" type="yyy"/>
```

where xxx is the name of the element and yyy is the data type of the element.

XML Schema has a lot of built-in data types. The most common types are:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Example

Simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Here are some XML elements:

```
<lastname>Ronald</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

Simple elements may have a default value which is automatically assigned to the element when no other value is specified as shown below:

```
<xs:element name="color" type="xs:string" default="red"/>
```

How to Define a Complex Element

We can define a complex element in an XML Schema two different ways:

1. The "employee" element can be declared directly by naming the element, like this:

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2. The "employee" element can have a type attribute that refers to the name of the complex type to use:

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

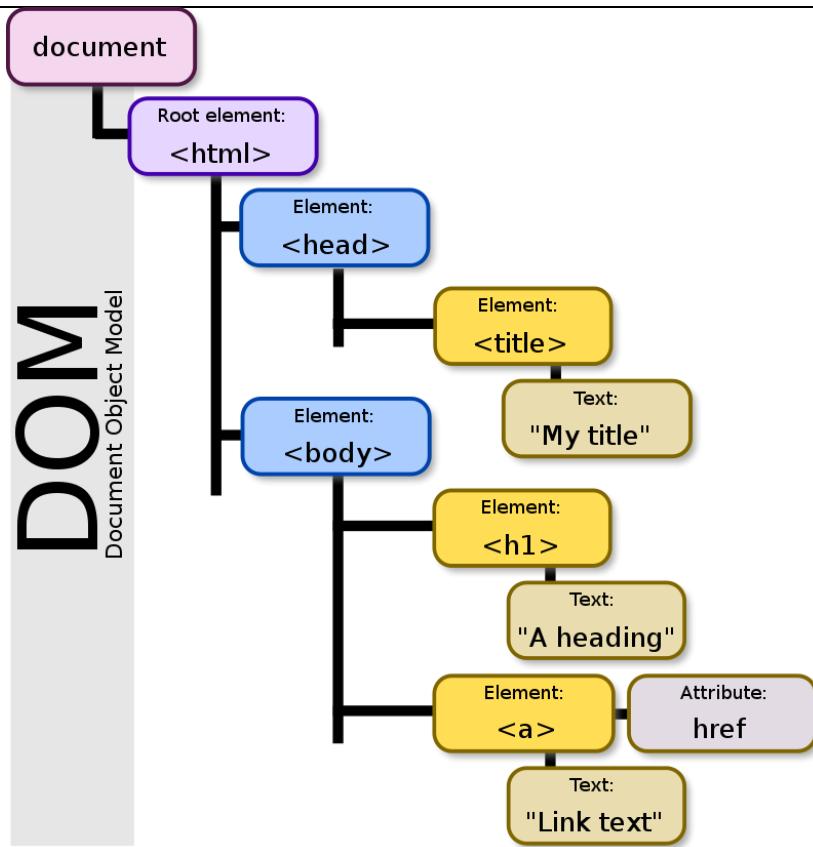
If you use the method described above, several elements can refer to the same complex type, like this:

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

(i) **Demonstrate** the building blocks of DOM.

9. The **Document Object Model (DOM)** is a [cross-platform](#) and [language](#)-independent interface that treats an [XML](#) or [HTML](#) document as a [tree structure](#) wherein each [node](#) is an [object](#) representing a part of the document. The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document. Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.



(ii) **Classify the types of DTD.**

XML DTD

An XML document with correct syntax is called "Well Formed".

An XML document validated against a DTD is both "Well Formed" and "Valid".

DTD stands for Document Type Definition.

A DTD defines the structure and the legal elements and attributes of an XML document.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

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

The DOCTYPE declaration above contains a reference to a DTD file. The content of the DTD file is shown and explained below.

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

Note.dtd:

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

The DTD above is interpreted like this:

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

Tip: #PCDATA means parseable character data.

How do you **infer the significant** differences between DID and XML schema for defining XML document structures with appropriate examples.

Difference Between XML Schema and DTD

10.

XML Schema vs. DTD

DTD, or Document Type Definition, and XML Schema, which is also known as XSD, are two ways of describing the structure and content of an XML document. DTD is the older

of the two, and as such, it has limitations that XML Schema has tried to improve.

Summary:

1. XML Schema is namespace aware, while DTD is not.
2. XML Schemas are written in XML, while DTDs are not.
3. XML Schema is strongly typed, while DTD is not.
4. XML Schema has a wealth of derived and built-in data types that are not available in DTD.
5. XML Schema does not allow inline definitions, while DTD does.

(i) List out data types data types of XML

XML Schema Data Types

XML Schema data types can be generally categorized a "simple type" (including embedded simple type) and "complex type." The "embedded simple type" is already defined, but can be used to create a new type through restriction or extension.

Table : XML Schema Data Types

11.	Simple Type	User can independently define. This type is used when a restriction is placed on an embedded simple type to create and use a new type.
	Complex Type	User can independently define. This type is used when the type has a child element or attribute.

A simple type is a type that only contains text data. This type can be used with element declarations and attribute declarations. On the other hand, a complex data type is a type that has a child element or attribute structure.

•Simple Type Example

```
<xs:element name="Department" type="xs:string" />
```

Here, the section described together with "xs:string" is an embedded simple type according to XML Schema. In this example, we have established the definition that the data type for the element called "Department" is a text string.

•Complex Type Example

```
<xs:complexType name="EmployeeType">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Name" />
    <xs:element ref="Department" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Name" type="xs:string" />
<xs:element name="Department" type="xs:string" />
```

(ii) **Explain** about the attributes of XML.

XML Elements vs. Attributes

Take a look at these examples:

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

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

In the first example gender is an attribute. In the last, gender is an element. Both examples provide the same information.

There are no rules about when to use attributes or when to use elements in XML.

Summarize on the following
(i) DOM based Parsing.

12.

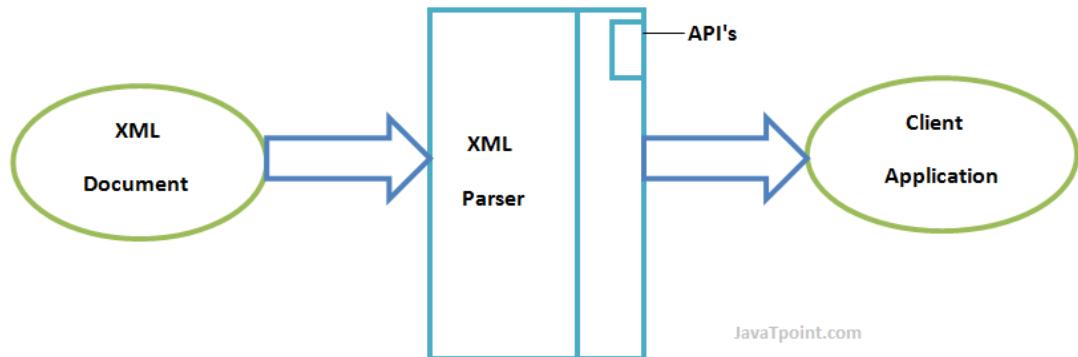
(ii) SAX based Parsing.

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

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

Let's understand the working of XML parser by the figure given below:



JavaTpoint.com

Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

DOM (Document Object Model)

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

Features of DOM Parser

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

DOM Parser has a tree based structure.

Advantages

- | | |
|--|--|
| | <ol style="list-style-type: none"> 1) It supports both read and write operations and the API is very simple to use. 2) It is preferred when random access to widely separated parts of a document is required. |
|--|--|

Disadvantages

- 1) It is memory inefficient. (consumes more memory because the whole XML document needs to be loaded into memory).
 - 2) It is comparatively slower than other parsers.
-

SAX (Simple API for XML)

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

Features of SAX Parser

It does not create any internal structure.

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

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

Advantages

- 1) It is simple and memory efficient.
- 2) It is very fast and works for huge documents.

Disadvantages

- 1) It is event-based so its API is less intuitive.
- 2) Clients never know the full information because the data is broken into pieces.

	(i) Compare and contrast RSS & ATOM.
--	---

 13. |

Difference Between RSS and ATOM

RSS vs ATOM

- Really Simple Syndication or RSS has been the standard for web feeds for a considerable time.
- Web feeds contains either a summary or the full text content of a web page.
- The problem with RSS is the often confusing and non standard conventions used by RSS due in part to its scattered development.
- The advent of the ATOM syndication standard was a response to the design flaws of the RSS standard.
- The primary advantage of the ATOM is its adaptation as the IETF standard.
- Being an IETF standard, each atom feed contains an explicit declaration of the format of the content along with what language is used.
- RSS feeds do not declare its content, but since it only contains plain text or escaped [HTML](#), it is rather easy for the browser to distinguish which is which.

A major flaw of RSS is in its code. RSS code isn't really very usable in other [XML](#) vocabularies since it wasn't really intended to do so at the very beginning. ATOM code has been built from the ground with modularity in mind. Therefore, a great majority of its code is reusable even with other XML vocabularies like RSS.

Summary:

1. ATOM is an IETF standard while RSS is not
2. ATOM feeds explicitly indicates the content while the browser is left to figure out whether the RSS feed contains plain text or escaped HTML
3. ATOM code is modular and reusable while RSS code is not
4. RSS still holds dominance in the syndication format due to its head start and popularity

(ii) **Explain** in detail about XSL elements.

XSLT <xsl:element>

Definition and Usage

The <xsl:element> element is used to create an element node in the output document.

Syntax

```
<xsl:element
  name="name"
  namespace="URI"
  use-attribute-sets="namelist">

  <!-- Content:template -->

</xsl:element>
```

Attributes

Attribute	Value	Description
name	name	Required. Specifies the name of the element to be created (the value of the name attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{\$country}" />)
namespace	URI	Optional. Specifies the namespace URI of the element (the value of the namespace attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{\$country}" namespace="{\$someuri}" />)
use-attribute-sets	namelist	Optional. A white space separated list of attribute-sets containing attributes to be added to the element

Example 1

Create a "singer" element that contains the value of each artist element:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <xsl:for-each select="catalog/cd">
      <xsl:element name="singer">
        <xsl:value-of select="artist" />
      </xsl:element>
      <br />
    </xsl:for-each>
  </xsl:template>

</xsl:stylesheet>
```

EXAMPLE OUTPUT FILES

XML File

```
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
</catalog>
```

XSL File

```
xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <body>
        <h2>My CD Collection</h2>
        <xsl:for-each select="catalog/cd">
```

```

<xsl:element name="singer">
<xsl:value-of select="artist"/>
</xsl:element>
<br/>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

RESULT

My CD Collection

Bob Dylan
 Bonnie Tyler
 Dolly Parton
 Gary Moore

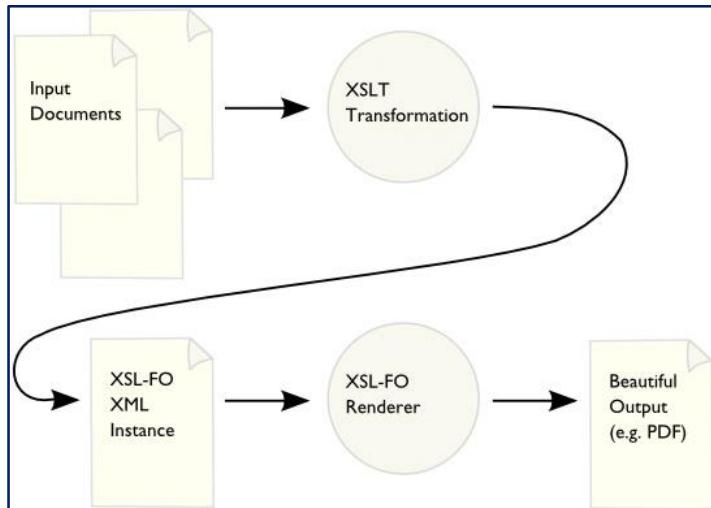
Explain in detail about
 (i) XSL and XSLT transformation

XSLT - Transformation

What is XSLT?

[XSL Transformations](#) (XSLT 2.0) is a language for transforming XML documents into other XML documents, text documents or HTML documents. You might want to format a chapter of a book using [XSL-FO](#), or you might want to take a database query and format it as HTML.

14.



Wildly Popular

XSLT has become the language of choice for a very wide range of XML applications. It is of course still used to produce XSL-FO documents for printing, but it is also used to integrate back-end software for Web sites. We can find XSLT inside most modern Web browsers, so

that XML can be transformed on the fly without the user even noticing; you will find XSLT on the desktop, in servers, in network appliances.

What is XSLT Used For?

If you make a purchase on eBay, or buy a book at Amazon, chances are that pretty much everything you see on every Web page has been processed with XSLT. Use XSLT to process multiple XML documents and to produce any combination of text, HTML and XML output. XSLT support is shipped with all major computer operating systems today, as well as being built in to all major Web browsers.

XSLT – Transformation : STEPS

- 1) Start with a Raw XML Document
- 2) Create an XSL Style Sheet
- 3) Link the XSL Style Sheet to the XML Document

1) Start with a Raw XML Document

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

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

2) Create an XSL Style Sheet

Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template:

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

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

  <xsl:template match="/">
    <html>
      <body>
        <h2>My CD Collection</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th>Title</th>
            <th>Artist</th>
          </tr>
```

```

<xsl:for-each select="catalog/cd">
<tr>
  <td><xsl:value-of select="title"/></td>
  <td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>

</xsl:stylesheet>

```

3) Link the XSL Style Sheet to the XML Document

Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>

    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
</catalog>

```

If you have an XSLT compliant browser it will nicely **transform** your XML into XHTML.

Sample OUTPUT

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore

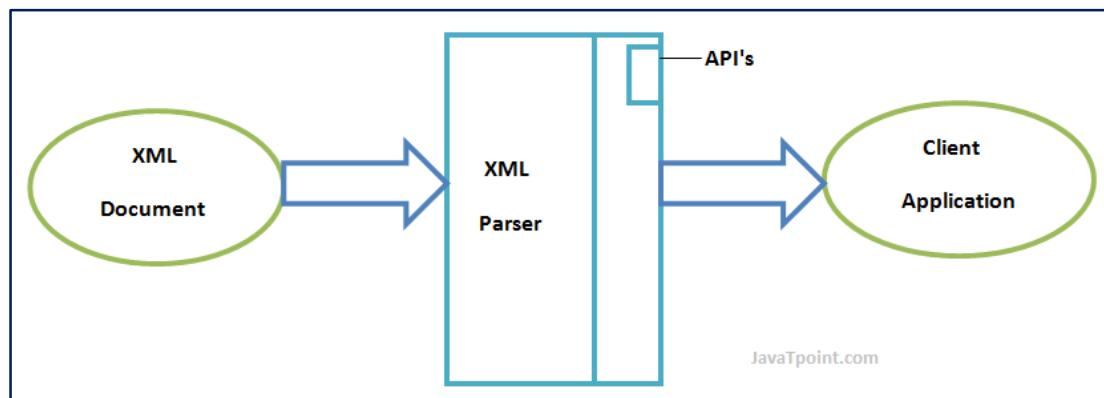
(ii) Comparison of DOM & SAX

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

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

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

These are the two main types of XML Parsers:

3. DOM
4. SAX

DOM (Document Object Model)

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

Features of DOM Parser

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

DOM Parser has a tree based structure.

Advantages

- 1) It supports both read and write operations and the API is very simple to use.
- 2) It is preferred when random access to widely separated parts of a document is required.

Disadvantages

- 1) It is memory inefficient. (consumes more memory because the whole XML document needs to be loaded into memory).
 - 2) It is comparatively slower than other parsers.
-

SAX (Simple API for XML)

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

Features of SAX Parser

It does not create any internal structure.

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

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

Advantages

- 1) It is simple and memory efficient.
- 2) It is very fast and works for huge documents.

Disadvantages

- 1) It is event-based so its API is less intuitive.
- 2) Clients never know the full information because the data is broken into pieces.

PART-C

1. **Explain how you shall carry out String Manipulations using a PHP Program.**

Manipulating PHP Strings

PHP provides many built-in functions for manipulating strings like calculating the length of a string, find substrings or characters, replacing part of a string with different characters, take a string apart, and many others.

Here are the examples of some of these functions.

Calculating the Length of a String

The `strlen()` function is used to calculate the number of characters inside a string. It also includes the blank spaces inside the string.

Example

Run this code »

```
<?php  
$my_str = 'Welcome to Tutorial Republic';  
  
// Outputs: 28  
echo strlen($my_str);  
?>
```

Counting Number of Words in a String

The `str_word_count()` function counts the number of words in a string.

Example

Run this code »

```
<?php  
$my_str = 'The quick brown fox jumps over the lazy dog.';  
  
// Outputs: 9  
echo str_word_count($my_str);  
?>
```

Replacing Text within Strings

The `str_replace()` replaces all occurrences of the search text within the target string.

Example

[Run this code »](#)

```
<?php  
$my_str = 'If the facts do not fit the theory, change the facts.';  
  
// Display replaced string  
echo str_replace("facts", "truth", $my_str);  
?>
```

The output of the above code will be:

If the truth do not fit the theory, change the truth.

You can optionally pass the fourth argument to the `str_replace()` function to know how many times the string replacements was performed, like this.

Example

[Run this code »](#)

```
<?php  
$my_str = 'If the facts do not fit the theory, change the facts.';  
  
// Perform string replacement  
str_replace("facts", "truth", $my_str, $count);  
  
// Display number of replacements performed  
echo "The text was replaced $count times.";  
?>
```

The output of the above code will be:

The text was replaced 2 times.

Reversing a String

The `strrev()` function reverses a string.

Example

[Run this code »](#)

```
<?php  
$my_str = 'You can do anything, but not everything.';  
  
// Display reversed string  
echo strrev($my_str);  
?>
```

	<p>The output of the above code will be:</p> <p>.gnihtyreve ton tub ,gnihtyna od nac uoY</p>								
2.	<p>Design a PHP application for College Management System with appropriate built-in functions and database.</p>								
	<p>Design application to send an email using PHP.</p> <h3><u>PHP mail() Function</u></h3> <h3>Example</h3> <p>Send a simple email:</p> <pre><?php // the message \$msg = "First line of text\nSecond line of text"; // use wordwrap() if lines are longer than 70 characters \$msg = wordwrap(\$msg,70); // send email mail("someone@example.com","My subject",\$msg); ?></pre>								
3.	<h3>Syntax</h3> <p><code>mail(<i>to,subject,message,headers,parameters</i>);</code></p> <h3>Parameter Values</h3> <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>to</i></td> <td>Required. Specifies the receiver / receivers of the email</td> </tr> <tr> <td><i>subject</i></td> <td>Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters</td> </tr> <tr> <td><i>message</i></td> <td>Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. Windows note: If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php</td> </tr> </tbody> </table>	Parameter	Description	<i>to</i>	Required. Specifies the receiver / receivers of the email	<i>subject</i>	Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters	<i>message</i>	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. Windows note: If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php
Parameter	Description								
<i>to</i>	Required. Specifies the receiver / receivers of the email								
<i>subject</i>	Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters								
<i>message</i>	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. Windows note: If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php								

		\$txt = str_replace("\n.", "\n..", \$txt); ?>	
<i>headers</i>		Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n). Note: When sending an email, it must contain a From header. This can be set with this parameter or in the php.ini file.	
<i>parameters</i>		Optional. Specifies an additional parameter to the sendmail program (the one defined in the sendmail_path configuration setting). (i.e. this can be used to set the envelope sender address when using sendmail with the -f sendmail option)	

Technical Details

Return Value:	Returns the hash value of the <i>address</i> parameter, or FALSE on failure. Note: Keep in mind that even if the email was accepted for delivery, it does NOT mean the email is actually sent and received!
PHP Version:	4+
PHP Changelog:	PHP 7.2: The headers parameter also accepts an array PHP 5.4: Added header injection protection for the <i>headers</i> parameter. PHP 4.3.0: (Windows only) All custom headers (like From, Cc, Bcc and Date) are supported, and are not case-sensitive. PHP 4.2.3: The <i>parameter</i> parameter is disabled in safe mode PHP 4.0.5: The <i>parameter</i> parameter was added

More Examples

Send an email with extra headers:

```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From: webmaster@example.com" . "\r\n" .
"CC: somebodyelse@example.com";

mail($to,$subject,$txt,$headers);
?>
```

Send an HTML email:

```
<?php
$to = "somebody@example.com, somebodyelse@example.com";
$subject = "HTML email";

$message = "
<html>
<head>
<title>HTML email</title>
</head>
<body>
<p>This email contains HTML Tags!</p>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
</tr>
</table>
</body>
</html>
";

// Always set content-type when sending HTML email
$headers = "MIME-Version: 1.0" . "\r\n";
$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

// More headers
$headers .= 'From: <webmaster@example.com>' . "\r\n";
$headers .= 'Cc: myboss@example.com' . "\r\n";

mail($to,$subject,$message,$headers);
?>
```

Summarize about XML schema and XML Parsers and Validation.

XML Schema

4. What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

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

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

Why Learn XML Schema?

In the XML world, hundreds of standardized XML formats are in daily use.

Many of these XML standards are defined by XML Schemas.

XML Schema is an XML-based (and more powerful) alternative to DTD.

XML Parser

All major browsers have a built-in XML parser to access and manipulate XML.

The [XML DOM \(Document Object Model\)](#) defines the properties and methods for accessing and editing XML.

However, before an XML document can be accessed, it must be loaded into an XML DOM object.

All modern browsers have a built-in XML parser that can convert text into an XML DOM object.

Parsing a Text String

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

Example

```
<html>
<body>

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

<script>
var text, parser, xmlDoc;

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

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

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

</body>
</html>
```

OUTPUT

Everyday Italian

XML - Validation

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration(DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are –

- Well-formed XML document
- Valid XML document

Well-formed XML Document

An XML document is said to be **well-formed** if it adheres to the following rules –

- Non DTD XML files must use the predefined character entities for **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)**.
- It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
- Each of its opening tags must have a closing tag or it must be a self ending tag.(<title>....</title> or <title/>).
- It must have only one attribute in a start tag, which needs to be quoted.
- **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)** entities other than these must be declared.

Example

Following is an example of a well-formed XML document –

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address
[
    <!ELEMENT address (name,company,phone)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT company (#PCDATA)>
    <!ELEMENT phone (#PCDATA)>
]>

<address>
    <name>Tanmay Patil</name>
    <company>TutorialsPoint</company>
    <phone>(011) 123-4567</phone>
</address>
```

The above example is said to be well-formed as –

- It defines the type of document. Here, the document type is **element** type.
- It includes a root element named as **address**.
- Each of the child elements among name, company and phone is enclosed in its self explanatory tag.
- Order of the tags is maintained.

Valid XML Document

If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.

CS8651 -UNIT 5-NOTES

CS8651- Internet Programming 2017Reg UNIT V - INTRODUCTION TO AJAX and WEB SERVICES	
PART-A	
No	Questions
1.	<p>Describe AJAX Control Extender Toolkit.</p> <p>What is the ASP.NET AJAX Control Toolkit?</p> <p>The ASP.NET AJAX Control Toolkit is an open-source project built on top of the Microsoft ASP.NET AJAX framework. It is a joint effort between Microsoft and the ASP.NET AJAX community that provides a powerful infrastructure to write reusable, customizable and extensible ASP.NET AJAX extenders and controls, as well as a rich array of controls that can be used out of the box to create an interactive Web experience.</p> <p>They are designed using concepts that are familiar to ASP.NET Web Forms application developers. Using the Ajax Control Toolkit, you can build Ajax-enabled ASP.NET Web Forms applications and ASP.NET MVC Web applications by dragging the controls from the Visual Studio Toolbox onto a page. The Ajax Control Toolkit is an open-source project that is part of the CodePlex Foundation</p> <p>The AJAX Control Toolkit contains more than 30 controls that enable you to easily create rich, interactive web pages.</p>
2.	<p>Discuss the advantages of AJAX.</p> <p><u>Advantages of AJAX</u></p> <ul style="list-style-type: none"> • Reduce the traffic travels between the client and the server. • Response time is faster so increases performance and speed. • You can use JSON (JavaScript Object Notation) which is alternative to XML. JSON is key value pair and works like an array. • You can use Firefox browser with an add-on called as Firebug to debug all Ajax calls. • Ready Open source JavaScript libraries available for use – JQuery, Prototype, Scriptaculous, etc.. • AJAX communicates over HTTP Protocol.
3.	<p>Identify the role of a callback function in performing a partial page update in an AJAX application.</p> <p><u>Partial-page rendering with UpdatePanels</u></p>

One of the most fascinating controls in the ASP.NET AJAX framework is the *UpdatePanel*. This new control replaces the need for a page to refresh during a postback. Only portions of a page, designated by the UpdatePanel, are updated. This technique is known as *partial-page rendering* and can be highly effective in improving the user experience.

6.1.1. Evolution of the UpdatePanel

For years, programming with the XMLHttpRequest object has been the most commonly used approach for communicating with the server from client-side script. The complexities involved in coding those types of applications scared away a lot of developers. To assist, the overall scripting model in ASP.NET 2.0 was significantly enhanced to introduce the idea of *script callbacks*—a way for server controls to communicate with client-side scripts between callbacks. This model was powerful because it offered access to the state of all the controls on the page during a callback. Unfortunately, many developers found the model difficult to work with, and numerous concerns were raised. The lack of support for passing complex types as parameters to the server (only strings were allowed) made the prototype too rigid and exposed its limitations. Developers began to look elsewhere for solutions.

In an effort to address these concerns, members of the ASP.NET team began work on a communication library built on top of the callbacks. The primary objective of the library was to simplify the use of callbacks and to provide a rich set of APIs for enabling the exchange of complex and simple types between the server and client. From this library came a control called the *RefreshPanel*. The purpose of the RefreshPanel was to offer a server control that refreshed the contents of a page without a page refresh. Out of this hard work, the UpdatePanel emerged, with deeper integration into the page lifecycle and a more transparent footprint on the page.

NOTE

A *callback* is a piece of code that is passed in as a parameter or argument to other code. The other piece of code can call the callback code (usually a function) at any time, even numerous times, in response to some processing.

4.	<p>Differentiate AJAX forms with HTML5 forms.</p> <p>AJAX is the name of a communication architecture between web pages and server side.</p> <p>JQuery is a javascript library that is written for unifying JS method calls (regarding DOM manipulation, String and Array functions, DOM queries...etc.) in all browsers.</p> <p>HTML5 is the rendering specification to be implemented by all browser providers. For this rendering to work JS Engine should also be updated, so HTML5 also means a JS engine with new features like drawing on a canvas.</p>										
5.	<p>What is XML Http Request object? List its properties.</p> <h3><u>The XMLHttpRequest Object</u></h3> <p>With the XMLHttpRequest object you can update parts of a web page, without reloading the whole page.</p> <h3>The XMLHttpRequest Object</h3> <p>The XMLHttpRequest object is used to exchange data with a server behind the scenes.</p> <p>The XMLHttpRequest object is the developers dream, because you can:</p> <ul style="list-style-type: none"> • Update a web page without reloading the page • Request data from a server after the page has loaded • Receive data from a server after the page has loaded • Send data to a server in the background <p>XMLHttpRequest Object Methods</p> <table border="1"> <thead> <tr> <th>Method</th><th>Description</th></tr> </thead> <tbody> <tr> <td>abort()</td><td>Cancels the current request</td></tr> <tr> <td>getAllResponseHeaders()</td><td>Returns header information</td></tr> <tr> <td>getResponseHeader()</td><td>Returns specific header information</td></tr> <tr> <td>open(method,url,async,uname,pswd)</td><td>Specifies the type of request, the URL, if the request should be handled asynchronously or not, and other optional attributes of a request</td></tr> </tbody> </table>	Method	Description	abort()	Cancels the current request	getAllResponseHeaders()	Returns header information	getResponseHeader()	Returns specific header information	open(method,url,async,uname,pswd)	Specifies the type of request, the URL, if the request should be handled asynchronously or not, and other optional attributes of a request
Method	Description										
abort()	Cancels the current request										
getAllResponseHeaders()	Returns header information										
getResponseHeader()	Returns specific header information										
open(method,url,async,uname,pswd)	Specifies the type of request, the URL, if the request should be handled asynchronously or not, and other optional attributes of a request										

		method: the type of request: GET or POST url: the location of the file on the server async: true (asynchronous) or false (synchronous)	
	send(string)	send(string) Sends the request off to the server. string: Only used for POST requests	
	setRequestHeader()	Adds a label/value pair to the header to be sent	

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number (e.g. "404" for "Not Found" or "200" for "OK")
statusText	Returns the status-text (e.g. "Not Found" or "OK")

6.	<p>Summarize the need of SOAP and show its structure.</p> <p><u>XML Soap</u></p> <ul style="list-style-type: none"> • SOAP stands for Simple Object Access Protocol • SOAP is an application communication protocol • SOAP is a format for sending and receiving messages • SOAP is platform independent
----	---

	<ul style="list-style-type: none"> • SOAP is based on XML • SOAP is a W3C recommendation <p>Why SOAP?</p> <p>It is important for web applications to be able to communicate over the Internet.</p> <p>The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.</p> <p>A SOAP message is an ordinary XML document containing the following elements:</p> <ul style="list-style-type: none"> • An Envelope element that identifies the XML document as a SOAP message • A Header element that contains header information • A Body element that contains call and response information • A Fault element containing errors and status information
7.	<p>Can you develop the service end point interface in RPC?</p> <p><i>Example</i></p> <p>Sample Java Beans service endpoint implementation and interface</p> <p>The following example illustrates a simple explicit Java Beans service endpoint implementation and the associated service endpoint interface.</p> <pre>/** This is an excerpt from the service implementation file, EchoServicePortTypeImpl.java package com.ibm.wssample.echo; import java.io.ByteArrayInputStream; import java.io.ByteArrayOutputStream; import javax.xml.bind.JAXBContext; import javax.xml.bind.Marshaller; import javax.xml.bind.Unmarshaller; import javax.xml.transform.stream.StreamSource;</pre> <pre>@javax.jws.WebService(serviceName = "EchoService", endpointInterface = "com.ibm.wssample.echo.EchoServicePortType", targetNamespace="http://com/ibm/was/wssample/echo/", portName="EchoServicePort") public class EchoServicePortTypeImpl implements EchoServicePortType { public EchoServicePortTypeImpl() { } public String invoke(String obj) { String str; str = obj; return str;</pre>

```

        }

}

/** This is a sample EchoServicePortType.java service interface */
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebResult;
import javax.jws.WebService;
import javax.xml.ws.*;

@WebService(name = "EchoServicePortType",
            targetNamespace =
"http://com.ibm/was/wssample/echo/",
            wsdlLocation="WEB-INF/wsdl/Echo.wsdl")
public interface EchoServicePortType {

    /** ...the method process ...*/
    @WebMethod
    @WebResult(name = "response", targetNamespace =
"http://com.ibm/was/wssample/echo/")
    @RequestWrapper(localName = "invoke", targetNamespace =
"http://com.ibm/was/wssample/echo/", className =
"com.ibm.was.wssample.echo.Invoke")
    @ResponseWrapper(localName = "echoStringResponse",
targetNamespace = "http://com.ibm/was/wssample/echo/", className =
"com.ibm.was.wssample.echo.EchoStringResponse")
    public String invoke(
        @WebParam(name = "arg0", targetNamespace =
"http://com.ibm/was/wssample/echo/")
        String arg0);

}

```

Sample Provider endpoint implementation

The following example illustrates a simple Provider service endpoint interface for a Java class.

```

package jaxws.provider.source;
import javax.xml.ws.Provider;
import javax.xml.ws.WebServiceProvider;
import javax.xml.transform.Source;

@WebServiceProvider() public class SourceProvider implements
Provider<Source> {

    public Source invoke(Source data) {
        return data;
    }
}

```

8. **List any four examples of web services.**
A Web Service Example

In the following example we will use ASP.NET to create a simple Web Service that converts the temperature from Fahrenheit to Celsius, and vice versa:

```
<%@ WebService Language="VBScript" Class="TempConvert" %>

Imports System
Imports System.Web.Services

Public Class TempConvert :Inherits WebService

<WebMethod()> Public Function FahrenheitToCelsius(ByVal
Fahrenheit As String) As String
    dim fahr
    fahr=trim(replace(Fahrenheit,",","."))
    if fahr="" or IsNumeric(fahr)=false then return "Error"
    return (((fahr) - 32) / 9) * 5
end function

<WebMethod()> Public Function CelsiusToFahrenheit(ByVal Celsius
As String) As String
    dim cel
    cel=trim(replace(Celsius,",","."))
    if cel="" or IsNumeric(cel)=false then return "Error"
    return (((cel) * 9) / 5) + 32
end function

end class
```

This document is saved as an .asmx file. This is the ASP.NET file extension for XML Web Services.

Put the Web Service on Your Web Site

Using a form and the HTTP POST method, you can put the web service on your site, like this:

Fahrenheit to Celsius:	<input type="text"/>	<input type="button" value="Submit"/>
------------------------	----------------------	---------------------------------------

Celsius to Fahrenheit:	<input type="text"/>	<input type="button" value="Submit"/>
------------------------	----------------------	---------------------------------------

code to add the Web Service to a web page:

```

<form action='tempconvert.asmx/FahrenheitToCelsius'
method="post" target="_blank">
<table>
  <tr>
    <td>Fahrenheit to Celsius:</td>
    <td>
      <input class="frmInput" type="text" size="30" name="Fahrenheit">
    </td>
  </tr>
  <tr>
    <td></td>
    <td align="right">
      <input type="submit" value="Submit" class="button">
    </td>
  </tr>
</table>
</form>

<form action='tempconvert.asmx/CelsiusToFahrenheit'
method="post" target="_blank">
<table>
  <tr>
    <td>Celsius to Fahrenheit:</td>
    <td>
      <input class="frmInput" type="text" size="30" name="Celsius">
    </td>
  </tr>
  <tr>
    <td></td>
    <td align="right">
      <input type="submit" value="Submit" class="button">
    </td>
  </tr>
</table>
</form>

```

Substitute the "tempconvert.asmx" with the address of your web service like:

<http://www.example.com/xml/tempconvert.asmx>

9. **Discover** an example for web service registry along with its functions.
- Web Services Discovery** provides access to software systems over the Internet using standard protocols. In the most basic scenario there is a *Web Service Provider* that publishes a service and a *Web Service Consumer* that uses this service. Web Service Discovery is the process of finding suitable [web services](#) for a given task.^[1]
- Publishing a web service involves creating a [software artifact](#) and making it accessible to potential consumers. Web service providers augment a [service endpoint](#)

	<p><u>interface</u> with an interface description using the <u>Web Services Description Language</u> (WSDL) so that a consumer can use the service.</p> <p>Universal Description, Discovery, and Integration (UDDI) is an XML-based registry for business internet services. A provider can explicitly register a service with a <i>Web Services Registry</i> such as UDDI or publish additional documents intended to facilitate discovery such as <u>Web Services Inspection Language</u> (WSIL) documents. The service users or consumers can search web services manually or automatically. The implementation of UDDI servers and WSIL engines should provide simple search APIs or web-based <u>GUI</u> to help find Web services.</p> <p>Web services may also be discovered using <u>multicast</u> mechanisms like <u>WS-Discovery</u>, thus reducing the need for centralized registries in smaller networks.</p>
10.	<p>Analyze the need for web service.</p> <p>We should use web services as it comes with various advantages listed below</p> <p>Re-usability</p> <p>Once we develop some business logic, we can make it reuse for other applications</p> <p>Example: If 10 different applications requires to use our logic We can expose our logic over a network as a <u>web service</u> So all the 10 different application can access it from the network.</p> <p>Interoperability</p> <p>It provides the freedom for a developers to choose whatever the technology they want to use for development.</p> <p>Web services uses a set of standards and protocols and enable us to achieve interoperability.</p> <p>Hence applications developed in Java, Mainframe, Ruby or any other technology can call the <u>web service</u> and use it.</p> <p>Loosely coupled</p> <p><u>Web service</u> exist independent of the other parts of the application that uses it. So any changes to the application can be made without affecting the web service.</p> <p>Deployability</p> <p>It is very easy to deploy the web application as they are deployed over standard internet technologies.</p>
11.	Give the uses of WSDL along with its definition.

WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

Features of WSDL

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

WSDL Usage

WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

12.	<p>Compare SOAP and HTTP.</p> <p>Difference between SOAP and HTTP</p> <table border="1" data-bbox="314 1358 1357 2021"><thead><tr><th data-bbox="314 1358 865 1426">SOAP</th><th data-bbox="865 1358 1357 1426">HTTP</th></tr></thead><tbody><tr><td data-bbox="314 1426 865 2021"><ul style="list-style-type: none">➤ SOAP was originally defined as S- Simple O- Object A-Access P-protocol.➤ It is a protocol specification which is used for exchanging structured information.➤ It is used in the implementation of web services in computer-based networks.➤ SOAP for its message format relies on XML Information set and sometimes relies on other application layer protocols as well, such as Hypertext Transfer Protocol (HTTP) or Simple</td><td data-bbox="865 1426 1357 2021"><ul style="list-style-type: none">➤ The HTTP or Hypertext Transfer Protocol (HTTP) is an application protocol which is used for distributed, collaborative and hypermedia information systems.➤ HTTP is widely regarded as the foundation of data communication for the World Wide Web (WWW).➤ Hypertext is a structured text that uses logical links or hyperlinks between those nodes that contain text. HTTP is the protocol</td></tr></tbody></table>	SOAP	HTTP	<ul style="list-style-type: none">➤ SOAP was originally defined as S- Simple O- Object A-Access P-protocol.➤ It is a protocol specification which is used for exchanging structured information.➤ It is used in the implementation of web services in computer-based networks.➤ SOAP for its message format relies on XML Information set and sometimes relies on other application layer protocols as well, such as Hypertext Transfer Protocol (HTTP) or Simple	<ul style="list-style-type: none">➤ The HTTP or Hypertext Transfer Protocol (HTTP) is an application protocol which is used for distributed, collaborative and hypermedia information systems.➤ HTTP is widely regarded as the foundation of data communication for the World Wide Web (WWW).➤ Hypertext is a structured text that uses logical links or hyperlinks between those nodes that contain text. HTTP is the protocol
SOAP	HTTP				
<ul style="list-style-type: none">➤ SOAP was originally defined as S- Simple O- Object A-Access P-protocol.➤ It is a protocol specification which is used for exchanging structured information.➤ It is used in the implementation of web services in computer-based networks.➤ SOAP for its message format relies on XML Information set and sometimes relies on other application layer protocols as well, such as Hypertext Transfer Protocol (HTTP) or Simple	<ul style="list-style-type: none">➤ The HTTP or Hypertext Transfer Protocol (HTTP) is an application protocol which is used for distributed, collaborative and hypermedia information systems.➤ HTTP is widely regarded as the foundation of data communication for the World Wide Web (WWW).➤ Hypertext is a structured text that uses logical links or hyperlinks between those nodes that contain text. HTTP is the protocol				

	<p>Mail Transfer Protocol (SMTP).</p> <ul style="list-style-type: none"> ➤ It is used for message negotiation and transmission mainly. ➤ SOAP forms the foundation layer of a web services protocol stack. 	<p>for exchanging or transferring hypertext.</p> <ul style="list-style-type: none"> ➤ The standards development of HTTP when it was innovated was coordinated by the Internet Engineering Task Force and the World Wide Web Consortium also called as W3C. 	
13.	<p>Summarize the need for enhancing security in web services.</p> <p>Definition - What does <i>Web Services Security (WS Security)</i> mean?</p> <p>Web Services Security (WS Security) is a specification that defines how security measures are implemented in web services to protect them from external attacks. It is a set of protocols that ensure security for SOAP-based messages by implementing the principles of confidentiality, integrity and authentication.</p> <p>Because Web services are independent of any hardware and software implementations, WS-Security protocols need to be flexible enough to accommodate new security mechanisms and provide alternative mechanisms if an approach is not suitable. Because SOAP-based messages traverse multiple intermediaries, security protocols need to be able to identify fake nodes and prevent data interpretation at any nodes. WS-Security combines the best approaches to tackle different security problems by allowing the developer to customize a particular security solution for a part of the problem. For example, the developer can select digital signatures for non-repudiation and Kerberos for authentication.</p>		
14.	<p>Name the types of indicators along with the definition.</p> <h3 style="color: red;">Web Services Security (WSS)</h3> <p>Web Services Security (WSS or WS-Security) describes enhancements to SOAP messaging in order to provide quality of protection through message integrity, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies.</p> <p>The scope of the Web Services Security Technical Committee is the support of security mechanisms in the following areas:</p>		

	<p>Using XML Signature to provide SOAP message integrity for Web Services</p> <p>Using XML Encryption to provide SOAP message confidentiality for Web Services</p> <p>Attaching and/or referencing security tokens in headers of SOAP messages. Options include:</p> <p>Username token</p> <p>SAML</p> <p>XrML</p> <p>Kerberos</p> <p>X.509</p> <p>Carrying security information for potentially multiple, designated actors</p> <p>Associating signatures with security tokens</p> <p>Each of the security mechanisms will use implementation and language neutral XML formats defined in XML Schema.</p>
	<p>Classify the basic concepts behind JAX-RPC technology.</p> <p>JAX-RPC</p> <p>Java APIs for XML-based Remote Procedure Call (JAX-RPC) help with Web service interoperability and accessibility by defining Java APIs that Java applications use to develop and access Web services. JAX-RPC fully embraces the heterogeneous nature of Web services -- it allows a JAX-RPC client to talk to another Web service deployed on a different platform and coded in a different language. Similarly, it also allows clients on other platforms and coded in different languages to talk to a JAX-RPC service. JAX-RPC also defines the mapping between WSDL service descriptions and Java interfaces.</p>
15.	<p>Th the JAX-RPC technology and describes its client and server programming models. JAX-RPC hides the complexity of underlying protocols and message-level processing from application developers crafting Web services using the Java 2 platform. The API combines XML with Remote Procedure Call (RPC), which is a mechanism enabling clients to execute procedures on distributed or remote systems, so that developers can build Web services and clients. The JAX-RPC remote procedure calls are represented by an XML infoset and they are carried over a network transport. While the JAX-RPC APIs rely on a XML-based protocol and a network transport, the APIs themselves are independent of a specific protocol or transport. The current JAX-RPC implementation relies on the SOAP 1.1 protocol and HTTP 1.1 network transport.</p>
16.	<p>What are the benefits of UDDI?</p> <p>Problems the UDDI specification can help to solve:</p> <ul style="list-style-type: none"> • Making it possible to discover the right business from the millions currently online • Defining how to enable commerce once the preferred business is discovered • Reaching new customers and increasing access to current customers • Expanding offerings and extending market reach

	<ul style="list-style-type: none"> • Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy • Describing services and business processes programmatically in a single, open, and secure environment
17.	<p>What are the core elements of UDDI?</p> <p>UDDI defines four core data elements within the data model:</p> <ul style="list-style-type: none"> • businessEntity (modeling business information) • businessService (describing a service) • tModel (describing specifications, classifications, or identifications) • binding Template (mapping between a businessService and the set of tModels that describe its technical fingerprint)
18.	<p>Rewrite the definition for UDDI.</p> <p>UDDI is an XML-based standard for describing, publishing, and finding web services.</p> <ul style="list-style-type: none"> • UDDI stands for Universal Description, Discovery, and Integration. • UDDI is a specification for a distributed registry of web services. • UDDI is a platform-independent, open framework. • UDDI can communicate via SOAP, CORBA, Java RMI Protocol. • UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services. • UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services. • UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet. <p>UDDI has two sections –</p> <ul style="list-style-type: none"> • A registry of all web service's metadata, including a pointer to the WSDL description of a service. • A set of WSDL port type definitions for manipulating and searching that registry.
19.	<p>Give the usage of UDDI in web service.</p> <p>UDDI is an XML-based standard for describing, publishing, and finding web services.</p> <ul style="list-style-type: none"> • UDDI stands for Universal Description, Discovery, and Integration. • UDDI is a specification for a distributed registry of web services. • UDDI is a platform-independent, open framework.

	<ul style="list-style-type: none"> • UDDI can communicate via SOAP, CORBA, Java RMI Protocol. • UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services. • UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services. • UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet. <p>UDDI has two sections –</p> <ul style="list-style-type: none"> • A registry of all web service's metadata, including a pointer to the WSDL description of a service. • A set of WSDL port type definitions for manipulating and searching that registry.
20.	<p>Define WSDL.</p> <p>WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.</p> <h3>Features of WSDL</h3> <ul style="list-style-type: none"> • WSDL is an XML-based protocol for information exchange in decentralized and distributed environments. • WSDL definitions describe how to access a web service and what operations it will perform. • WSDL is a language for describing how to interface with XML-based services. • WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry. • WSDL is the language that UDDI uses. • WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'. <h3>WSDL Usage</h3> <p>WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.</p>
	PART-B
1.	(i) Describe in detail about the AJAX architecture.

	<p>(ii) List out the call back methods.</p>
	<p>(i) Analyze various concepts of RPC.</p> <h2>Remote Procedure Call (RPC)</h2> <p>A remote procedure call is an interprocess communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call.</p> <p>A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumes execution after the server is finished.</p> <p>The sequence of events in a remote procedure call are given as follows:</p> <ul style="list-style-type: none"> • The client stub is called by the client. • The client stub makes a system call to send the message to the server and puts the parameters in the message. • The message is sent from the client to the server by the client's operating system. • The message is passed to the server stub by the server operating system. • The parameters are removed from the message by the server stub. • Then, the server procedure is called by the server stub. <p>2. A diagram that demonstrates this is as follows:</p> <pre> graph LR subgraph Client [Client] direction TB CF[Client Functions] --> CS[Client Stub] CS --> RR[RPC Runtime] RR --> SF[Server Functions] SF --> SS[Server Stub] SS --> RR end subgraph Server [Server] direction TB SF[Server Functions] --> SS[Server Stub] SS --> RR[RPC Runtime] RR --> CF[Client Functions] end </pre> <p>The diagram illustrates the RPC architecture between a Client and a Server. Both sides have a layered structure: Client Functions at the top, followed by a Client Stub, and finally an RPC Runtime layer at the bottom. On the Server side, the layers are Server Functions, Server Stub, and RPC Runtime. Arrows indicate the flow of control and data: Client Functions call the Client Stub, which calls the RPC Runtime, which then calls the Server Functions. The Server Functions call the Server Stub, which calls the RPC Runtime, which finally calls the Client Functions. There are also feedback arrows from the Client Functions back to the Client Stub and from the Server Functions back to the Server Stub.</p>

Advantages of Remote Procedure Call

Some of the advantages of RPC are as follows:

- Remote procedure calls support process oriented and thread oriented models.
- The internal message passing mechanism of RPC is hidden from the user.
- The effort to re-write and re-develop the code is minimum in remote procedure calls.
- Remote procedure calls can be used in distributed environment as well as the local environment.
- Many of the protocol layers are omitted by RPC to improve performance.

Disadvantages of Remote Procedure Call

Some of the disadvantages of RPC are as follows:

- The remote procedure call is a concept that can be implemented in different ways. It is not a standard.
- There is no flexibility in RPC for hardware architecture. It is only interaction based.
- There is an increase in costs because of remote procedure call.

(ii) **Explain** the basic concepts behind JAX-RPC.

JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.

In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. In this release, JAX-RPC relies on SOAP 1.1 and HTTP 1.1.

Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer. On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy.

With JAX-RPC, clients and Web services have a big advantage--the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide

	<p>Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.</p>
3.	<p>Explain in detail with an example of Java Web Services.</p> <p>With a simple example illustrate the steps to create a java web service. (NOV/DEC 2012)</p> <p>Writing a java web service</p> <p>Currency conversion Service</p> <ul style="list-style-type: none"> ◆ Writing a server for a service using JWSDP 1.3 tools ◆ Application: currency converter ■ Three operations: ● fromDollars ● fromEuros ● fromYen ■ Input: value in specified currency ■ Output: object containing input value and equivalent values in other two currencies <p>Writing server software</p> <ol style="list-style-type: none"> 1. Write service endpoint interface • May need to write additional classes representing data structures 2. Write class implementing the interface 3. Compile classes 4. Create configuration files and run JWSDP tools to create web service 5. Deploy web service to Tomcat <p>service endpoint interface</p> <ul style="list-style-type: none"> ◆ The Web service endpoint interface is used to define the ‘Web services methods’. ◆ A Web service endpoint interface must conform to the rules of a JAX-RPC service definition interface. ◆ a service endpoint interface (SEI) that defines the interface of the web service. ◆ Configuration files are XML files that can be changed as needed. Developers can use configuration files to change settings without recompiling applications. <p>Administrators can use configuration files to set policies that affect how applications run on their computers.</p> <ul style="list-style-type: none"> ◆ config.xml : Defines the URL for WSDL file location. Each Web services has a corresponding WSDL (Web service Definition Language) document. <p>JWSDP: Server</p> <p>Rules for Service endpoint interface</p> <ul style="list-style-type: none"> ■ Must extend java.rmi.Remote ● declares a set of methods that may be invoked from a remote Java Virtual Machine(JVM) ■ Every method must throw java.rmi.RemoteException

- Parameter/return value data types are restricted
- No public static final declarations (global constants) It must not have constant declarations
 - ◆ **Allowable parameter/return value data types**
 - Java primitives (int, boolean, *etc.*)
 - Primitive wrapper classes (Integer, *etc.*)
 - String, Date, Calendar, BigDecimal, BigInteger
 - java.xml.namespace.QName, java.net.URI
 - Struct: class consisting entirely of public instance variables
 - Array of any of the above
 - ◆ **Struct for currency converter app (data type for return values)**

```
package myCurCon;

public class ExchangeValues {
    public double dollars;
    public double euros;
    public double yen;
}
```

◆ **Service endpoint interface**

```
package myCurCon;

public interface CurCon extends java.rmi.Remote {
    public ExchangeValues fromDollars(double dollars)
        throws java.rmi.RemoteException;
    public ExchangeValues fromEuros(double euros)
        throws java.rmi.RemoteException;
    public ExchangeValues fromYen(double yen)
        throws java.rmi.RemoteException;
}

◆ Three files ExchangeValues.java, CurCon.java and CurConImpl.java written for the web service
◆ Class CurConImpl contains methods implements service endpoint interface, for example:
public ExchangeValues fromDollars(double dollars)
    throws java.rmi.RemoteException
{
    ExchangeValues ev = new ExchangeValues();
    ev.dollars = dollars;
    ev.euros = dollars * dollar2euro;
    ev.yen = dollars * dollar2yen;
    return ev;
}
```

Packaging server software

- ◆ Configuration file input to wscompile to create server

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <service
    name="HistoricCurrencyConverter"
    targetNamespace="http://tempuri.org/wsdl"
    typeNamespace="http://tempuri.org/types"
    packageName="myCurCon">
    <interface name="myCurCon.CurCon" />
  </service>
</configuration>
```

- ◆ Configuration file for web service

```
<?xml version="1.0" encoding="UTF-8"?>
<webServices
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/dd"
  version="1.0"
  targetNamespaceBase="http://tempuri.org/wsdl"
  typeNamespaceBase="http://tempuri.org/types"
  >
```

- ◆ Configuration file for web service

```
  urlPatternBase="/converter">
```

Context path

```
<endpoint
  name="CurrConverter"
  displayName="Currency Converter"
  description=
    "Converts between dollars, euros, and yen."
  interface="myCurCon.CurCon"
  model="/WEB-INF/model.xml.gz"
  implementation="myCurCon.CurConImpl"/>
```

Like servlet in web.xml

```
<endpointMapping
  endpointName="CurrConverter"
  urlPattern="/currency" />
```

Like servlet-mapping in web.xml

```
</webServices>
```

- ◆ Also need a minimal web.xml

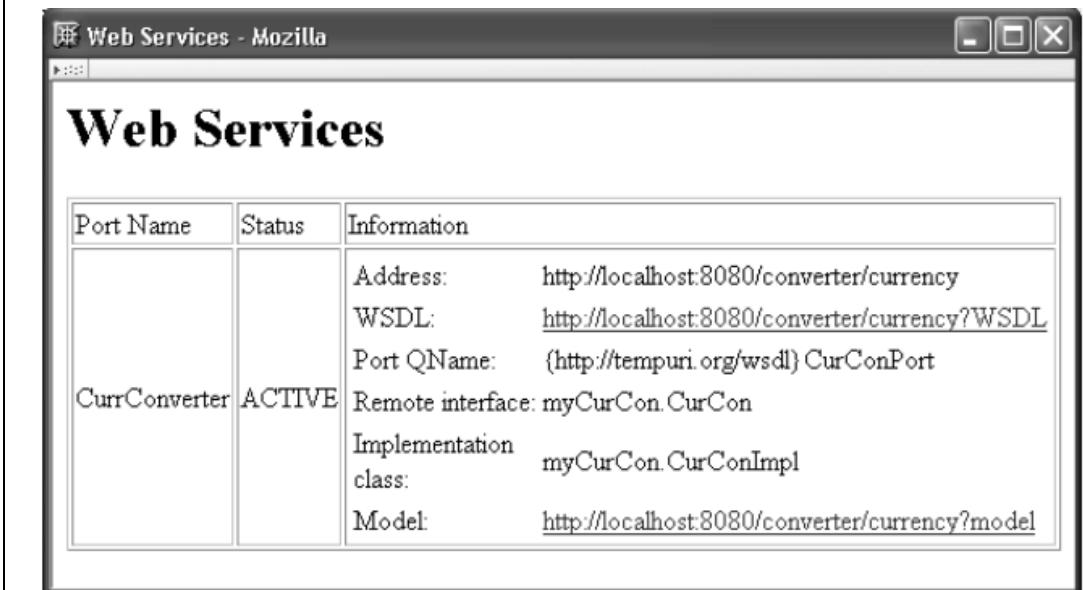
```
<web-app
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">
```

```
  <display-name>Historic Currency Converter</display-name>
  <description>
    This web service converts between three currencies using their
    exchange rates as of a fixed date.
  </description>
</web-app>
```

Run jar and wsdeploy to create a Web Archive (WAR) file converter.war

- Name must match urlPatternBase value

- ◆ `jaxrpc-ri.xml`: Defines the various end points for referencing a Web service.
- ◆ `wscompile`: The wscompile tool generates stubs, and WSDL files used in JAX-RPC clients and services. The tool reads as input a configuration file and either a WSDL file or an RMI interface that defines the service.
- ◆ `wsdeploy`: Reads a WAR file (something like Jar file) and the `jaxrpc-ri.xml` file and then generates another WAR file that is ready for deployment
 - ◆ Write service endpoint interface
 - May need to write additional classes representing data structures
 - ◆ Write class implementing the interface
 - ◆ Compile classes
 - ◆ Create configuration files and run JWSDP tools to create web service
 - ◆ Deploy web service to Tomcat
 - ◆ Just copy converter.war to Tomcat webapps directory
 - May need to use Manager app to deploy
 - Enter converter.war in “WAR or Directory URL” text box
 - ◆ Testing success:
 - Visit <http://localhost:8080/converter/currency>



Discuss in detail the architecture of web services.

Architecture of Web Services

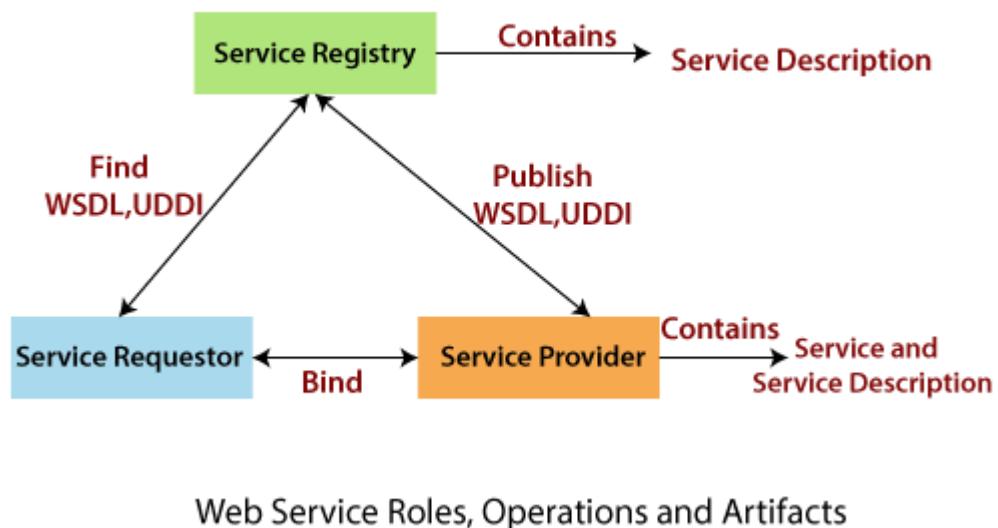
4. The Web Services architecture describes how to instantiate the elements and implement the operations in an interoperable manner.

The architecture of web service interacts among three roles: **service provider**, **service requester**, and **service registry**. The interaction involves the three operations: **publish**, **find**, and **bind**. These operations and roles act upon

the **web services artifacts**. The web service artifacts are the web service software module and its description.

The service provider hosts a network-associable module (web service). It defines a service description for the web service and publishes it to a service requestor or service registry. These service requestor uses a find operation to retrieve the service description locally or from the service registry. It uses the service description to bind with the service provider and invoke with the web service implementation.

The following figure illustrates the operations, roles, and their interaction.



Roles in a Web Service Architecture

There are three roles in web service architecture:

- Service Provider
- Service Requestor
- Service Registry

Service Provider

From an architectural perspective, it is the platform that hosts the services.

Service Requestor

Service requestor is the application that is looking for and invoking or initiating an interaction with a service. The browser plays the requester role, driven by a consumer or a program without a user interface.

Service Registry

Service requestors find service and obtain binding information for services during development.

Operations in a Web Service Architecture

Three behaviors that take place in the microservices:

- Publication of service descriptions (**Publish**)
- Finding of services descriptions (**Find**)
- Invoking of service based on service descriptions (**Bind**)

Publish: In the publish operation, a service description must be published so that a service requester can find the service.

Find: In the find operation, the service requestor retrieves the service description directly. It can be involved in two different lifecycle phases for the service requestor:

- At design, time to retrieve the service's interface description for program development.
- And, at the runtime to retrieve the service's binding and location description for invocation.

Bind: In the bind operation, the service requestor invokes or initiates an interaction with the service at runtime using the binding details in the service description to locate, contact, and invoke the service.

Artifacts of the web service

There are two artifacts of web services:

- Service
- Service Registry

Service: A service is an **interface** described by a service description. The service description is the implementation of the service. A service is a software module deployed on network-accessible platforms provided by the service provider. It interacts with a service requestor. Sometimes it also functions as a requestor, using other Web Services in its implementation.

Service Description: The service description comprises the details of the **interface** and **implementation** of the service. It includes its **data types**, **operations**, **binding information**, and **network location**. It can also categorize other metadata to enable discovery and utilize by service requestors. It can be published to a service requestor or a service registry.

5. (i) Deduce any two elements of WSDL.

WSDL (Web services description language)

A web service cannot be used if it cannot be found. The client invoking the web service should know where the web service actually resides.

Secondly, the client application needs to know what the web service actually does, so that it can invoke the right web service. This is done with the help of the WSDL, known as the Web services description language. The WSDL file is again an XML-based file which basically tells the client application what the web service does. By using the WSDL document, the client application would be able to understand where the web service is located and how it can be utilized.

Web Service Example

An example of a WSDL file is given below.

```
<definitions>
    <message name="TutorialRequest">
        <part name="TutorialID" type="xsd:string"/>
    </message>

    <message name="TutorialResponse">
        <part name="TutorialName" type="xsd:string"/>
    </message>

    <portType name="Tutorial_PortType">
        <operation name="Tutorial">
            <input message="tns:TutorialRequest"/>
            <output message="tns:TutorialResponse"/>
        </operation>
    </portType>

    <binding name="Tutorial_Binding" type="tns:Tutorial_PortType">
        <soap:binding style="rpc"
                      transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="Tutorial">
            <soap:operation soapAction="Tutorial"/>
            <input>
                <soap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
            </input>
            <output
                namespace="urn:examples:Tutorialservice"/>
        </operation>
    </binding>
</definitions>
```

```

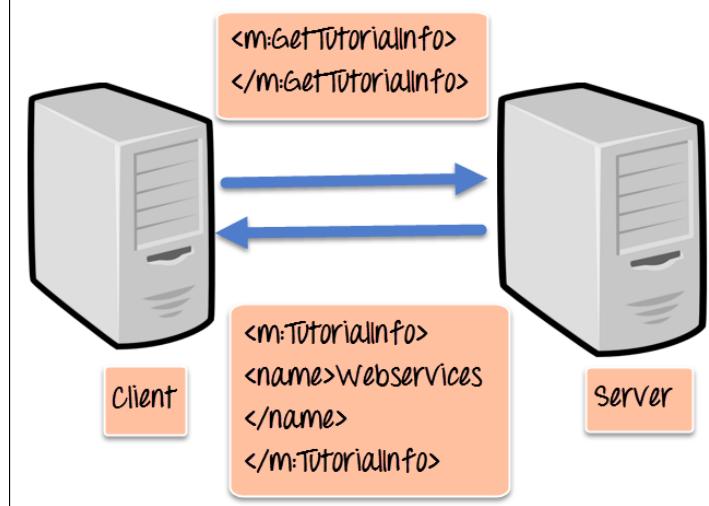
        use="encoded"/>
    </input>

    <output>
        <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:Tutorialservice"
            use="encoded"/>
    </output>
</operation>
</binding>
</definitions>
```

(ii) **Explain** the steps for writing web service.

6.	<p>Describe briefly about the elements of WSDL.</p> <p>The important aspects to note about the above WSDL declaration are as follows;</p> <ol style="list-style-type: none"> 1. <message> - The message parameter in the WSDL definition is used to define the different data elements for each operation performed by the web service. So in the example above, we have 2 messages which can be exchanged between the web service and the client application, one is the "TutorialRequest", and the other is the "TutorialResponse" operation. The TutorialRequest contains an element called "TutorialID" which is of the type string. Similarly, the TutorialResponse operation contains an element called "TutorialName" which is also a type string. 2. <portType> - This actually describes the operation which can be performed by the web service, which in our case is called Tutorial. This operation can take 2 messages; one is an input message, and the other is the output message. 3. <binding> - This element contains the protocol which is used. So in our case, we are defining it to use http (http://schemas.xmlsoap.org/soap/http). We also specify other details for the body of the operation, like the
----	--

	<p>namespace and whether the message should be encoded.</p>
7.	<p>(i) Summarize on the structure of SOAP.</p> <h3 style="text-align: center;">SOAP (Simple Object Access Protocol)</h3> <p>SOAP is known as a transport-independent messaging protocol. SOAP is based on transferring XML data as SOAP Messages. Each message has something which is known as an XML document. Only the structure of the XML document follows a specific pattern, but not the content. The best part of Web services and SOAP is that its all sent via HTTP, which is the standard web protocol.</p> <p>Here is what a SOAP message consists of</p> <ul style="list-style-type: none"> ○ Each SOAP document needs to have a root element known as the <Envelope> element. The root element is the first element in an XML document. ○ The "envelope" is in turn divided into 2 parts. The first is the header, and the next is the body. ○ The header contains the routing data which is basically the information which tells the XML document to which client it needs to be sent to. ○ The body will contain the actual message. <p>The diagram below shows a simple example of the communication via SOAP.</p>



(ii) **Describe** briefly about SOAP & HTTP.

(i) **Demonstrate** the building blocks of SOAP.

XML Soap

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation

8. Why SOAP?

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

(ii) **Classify** the encoding of struct data and array.

SOAP - Encoding

SOAP includes a built-in set of rules for encoding data types. It enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

- SOAP data types are divided into two broad categories – scalar types and compound types.
- Scalar types contain exactly one value such as a last name, price, or product description.
- Compound types contain multiple values such as a purchase order or a list of stock quotes.
- Compound types are further subdivided into arrays and structs.

Compound Types

SOAP arrays have a very specific set of rules, which require that you specify both the element type and array size. SOAP also supports multidimensional arrays, but not all SOAP implementations support multidimensional functionality.

To create an array, you must specify it as an `xsi:type` of array. The array must also include an `arrayType` attribute. This attribute is required to specify the data type for the contained elements and the dimension(s) of the array.

For example, the following attribute specifies an array of 10 double values –

```
arrayType = "xsd:double[10]"
```

In contrast, the following attribute specifies a two-dimensional array of strings –

```
arrayType = "xsd:string[5,5]"
```

Here is a sample SOAP response with an array of double values –

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
    envelope"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:getPriceListResponse
            xmlns:ns1 = "urn:examples:pricelistservice"
            SOAP-ENV:encodingStyle =
            "http://www.w3.org/2001/12/soap-encoding">

            <return xmlns:ns2 =
            "http://www.w3.org/2001/09/soap-encoding"
            xsi:type = "ns2:Array" ns2:arrayType =
            "xsd:double[2]">
                <item xsi:type = "xsd:double">54.99</item>
                <item xsi:type = "xsd:double">19.99</item>
            </return>
        </ns1:getPriceListResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Structs contain multiple values, but each element is specified with a unique accessor element. For example, consider an item within a product catalog. In this case, the struct might contain a product SKU, product name, description, and price. Here is how such a struct would be represented in a SOAP message –

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
    envelope"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

    <SOAP-ENV:Body>
        <ns1:getProductResponse
            xmlns:ns1 = "urn:examples:productservice">
```

```

SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">

    <return xmlns:ns2 = "urn:examples" xsi:type =
"ns2:product">
        <name xsi:type = "xsd:string">Red Hat
Linux</name>
        <price xsi:type = "xsd:double">54.99</price>
        <description xsi:type = "xsd:string">
            Red Hat Linux Operating System
        </description>
        <SKU xsi:type = "xsd:string">A358185</SKU>
    </return>
</ns1:getProductResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Analyze the various steps in database driven web service with some example.

Overview of Database Web Services

Web services enable application-to-application interaction over the Web, regardless of platform, language, or data formats. The key ingredients, including Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI), have been adopted across the entire software industry. Web services usually refer to services implemented and deployed in middle-tier application servers. However, in heterogeneous and disconnected environments, there is an increasing need to access stored procedures, as well as data and metadata, through Web services interfaces.

The Database Web services technology is a database approach to Web services. It works in the following two directions:

9.

- Accessing database resources as a Web service
- Consuming external Web services from the database

Oracle Database can access Web services through PL/SQL packages and Java classes deployed within the database. Turning Oracle Database into a Web service provider leverages investment in Java stored procedures, PL/SQL packages, predefined SQL queries, and data manipulation language (DML). Conversely, consuming external Web services from the database, together with integration with the SQL engine, enables Enterprise Information Integration.

Using Oracle Database as Web Services Provider

Web Services use industry-standard mechanisms to provide easy access to remote content and applications, regardless of the platform and location of the provider and implementation and data format. Client applications can query and retrieve data from Oracle Database and call stored procedures using standard Web service protocols. There is no dependency on Oracle-specific

	<p>database connectivity protocols. This approach is highly beneficial in heterogeneous, distributed, and disconnected environments.</p> <p>You can call into the database from a Web service, using the database as a service provider. This enables you to leverage existing or new SQL, PL/SQL, Java stored procedures, or Java classes within Oracle Database. You can access and manipulate database tables from a Web service client.</p>
	<p>Illustrate on web services for writing web service client along with the description of WSDL.</p> <h2>Creating a Web Service Client</h2> <p>Creating a web service client application always starts with an existing WSDL file.</p> <p>Typically, you retrieve the WSDL directly from a web service provider using the <code>wsimport</code> tool. The <code>wsimport</code> tool then generates the corresponding Java source code for the interface described by the WSDL. The Java compiler, <code>javac</code>, is then called to compile the source into class files. The programming code uses the generated classes to access the web service.</p> <h3>Creating a Client from WSDL</h3> <p>To create a client from WSDL, you must create the following files:</p> <ul style="list-style-type: none"> • Client Java File (fromwsdl) • Client Configuration File (fromwsdl) • <code>build.xml</code> • <code>build.properties</code> <p>Client Java File (fromwsdl)</p> <p>The client Java file defines the functionality of the web service client. The following code shows the <code>AddNumbersClient.java</code> file that is provided in the sample.</p> <pre> package fromjava.client; import com.sun.xml.ws.Closeable; import java.rmi.RemoteException; public class AddNumbersClient { public static void main (String[] args) { AddNumbersImpl port = null; try { port = new AddNumbersImplService().getAddNumbersImplPort(); int number1 = 10; int number2 = 20; System.out.printf ("Invoking addNumbers(%d, %d)\n", number1, number2); int result = port.addNumbers (number1, number2); System.out.printf ("The result of adding %d and %d is %d.\n\n", number1, number2, result); number1 = -10; } } </pre>
10.	

```

        System.out.printf ("Invoking addNumbers(%d, %d)\n",
                           number1, number2);
        result = port.addNumbers (number1, number2);
        System.out.printf (
            "The result of adding %d and %d is %d.\n",
            number1, number2, result);
    } catch (AddNumbersException_Exception ex) {
        System.out.printf (
            "Caught AddNumbersException_Exception: %s\n",
            ex.getFaultInfo ().getDetail ());
    } finally {
        ((Closeable)port).close();
    }
}
}

```

This file specifies two positive integers that are to be added by the web service, passes the integers to the web service and gets the results from the web service via the `port.addNumbers` method, and prints the results to the screen. It then specifies a negative number to be added, gets the results (which should be an exception), and prints the results (the exception) to the screen.

Client Configuration File (fromwsdl)

This is a sample `custom-client.xml` file. The `wsdlLocation`, package name, and `jaxb:package` name `xml` tags are unique to each client and are highlighted in bold text

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bindings
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    wsdlLocation="http://localhost:8080/wsit-enabled-fromwsdl/
        addnumbers?wsdl"
    xmlns="http://java.sun.com/xml/ns/jaxws">
    <bindings node="ns1:definitions"
        xmlns:ns1="http://schemas.xmlsoap.org/wsdl/">
        <package name="fromwsdl.client" />
    </bindings>
    <bindings node="ns1:definitions/ns1:types/xsd:schema
        [@targetNamespace='http://duke.org']"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:ns1="http://schemas.xmlsoap.org/wsdl/">
        <jaxb:schemaBindings>
            <jaxb:package name="fromwsdl.client" />
        </jaxb:schemaBindings>
    </bindings>
</bindings>

```

- | | |
|--|--|
| | <p>(i) List out the installation steps of JWSDP.</p> <p>JWSDP</p> <ol style="list-style-type: none"> 1. Install JDK 6.0 (i.e., JDK 1.6.0) <p>Set up the following environment variables:</p> <p>JAVA_HOME C:\Program Files\Java\jdk1.6.0_07</p> <p>Add the following path:</p> <p>C:\Program Files\Java\jdk1.6.0_07\bin</p> |
|--|--|

2. Install [JWSDP 2.0](#) & [Tomcat 5.0 for JWSDP](#) (based upon Tomcat 5.0.19 that implements the Java Server Pages 2.0 and Java Servlet 2.4 specifications)

Set up the following environment variables:
 JWSDP_HOME C:\Sun\jwsdp-2.0
 ANT_HOME C:\Sun\jwsdp-2.0\apache-ant

Add the following path:
 C:\Sun\jwsdp-2.0\jwsdp-shared\bin;C:\Sun\jwsdp-2.0\apache-ant\bin

3. Copy [examples.zip](#) into C:\ and extract here

4. Copy [lib.zip](#) into C:\Sun\jwsdp-2.0\server directory and extract here
 Delete the file "lib.zip"

5. Replace saaj-impl.jar file at the following directories by [saaj-impl-1.3.jar](#).
 Rename it to saaj-impl.jar.

C:\Sun\jwsdp-2.0\saaj\lib
 C:\Sun\tomcat50-jwsdp\saaj\lib

6. Modify C:\examples\common\build.properties for the first four lines as follows:

```
tutorial.home=C:
tutorial.install=${tutorial.home}
username=hxu
password=12345
```

where "hxu" and "12345" are the username and password for the Tomcat server.

7. Build server:
 cd C:\examples\jaxrpc\helloservice
 ant build

Start Tomcat from JWSDP 2.0

Deploy server:
 ant deploy

Note: If application already exists at path /hello-jaxrpc, you should use the command "ant undeploy" to undeploy the web service first.

Verify the deployment:
 To verify that the service has been successfully deployed, open a browser window and specify the service endpoint's URL as follows:

<http://localhost:8080/hello-jaxrpc/hello?WSDL>

You should get the following display.

```

<?xml version="1.0" encoding="UTF-8"?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="MyHelloService" targetNamespace="urn:Foo">
  <types />
  - <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string" />
  </message>
  + <message name="HelloIF_sayHelloResponse">
  - <portType name="HelloIF">
    - <operation name="sayHello" parameterOrder="String_1">
      <input message="tns:HelloIF_sayHello" />
      <output message="tns:HelloIF_sayHelloResponse" />
    </operation>
  </portType>
  - <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
    - <operation name="sayHello">
      <soap:operation soapAction="" />
      - <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="urn:Foo" />
      </input>
      - <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded" namespace="urn:Foo" />
      </output>
    </operation>
  </binding>
  - <service name="MyHelloService">
    - <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address location="http://localhost:8080/hello-jaxrpc/hello" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
    </port>
  </service>
</definitions>

```

8. Build client:

```
cd C:\examples\jaxrpc\dynamicproxy
ant build
```

Run client:
ant run

```

C:\ Command Prompt
[echo] Creating the required directories.....
package-dynamic:
[echo] Building the client JAR file....
[delete] Deleting: C:\examples\jaxrpc\dynamicproxy\dist\client.jar
[jar] Building jar: C:\examples\jaxrpc\dynamicproxy\dist\client.jar

build-dynamic:

build:

BUILD SUCCESSFUL
Total time: 20 seconds
C:\examples\jaxrpc\dynamicproxy>ant run
Buildfile: build.xml

run-client:
[javal UrlString = http://localhost:8080/hello-jaxrpc/hello?WSDL
[javal Hello Buzz

run:

BUILD SUCCESSFUL
Total time: 3 seconds
C:\examples\jaxrpc\dynamicproxy>

```

(ii) **Describe** on Simple Object Access Protocol.

XML Soap

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation

Why SOAP?

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

12.	(i) Discuss the XMLHttpRequest Object with example.
-----	--

AJAX - The XMLHttpRequest Object

The keystone of AJAX is the XMLHttpRequest object.

The XMLHttpRequest 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 web page, without reloading the whole page.

Create an 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();
```

Example

```
var xhttp = new XMLHttpRequest();
```

Access Across Domains

For security reasons, modern browsers do not allow access across domains.

This means that both the web page and the XML file it tries to load, must be located on the same server.

The examples on W3Schools all open XML files located on the W3Schools domain.

If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.

Example

```
if (window.XMLHttpRequest) {  
    // code for modern browsers  
    xmlhttp = new XMLHttpRequest();  
} else {  
    // code for old IE browsers
```

```
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

(ii) **Describe** about Java web service basics.

JAX-WS Example RPC Style

Creating JAX-WS example is a easy task because it requires no extra configuration settings.

JAX-WS API is inbuilt in JDK, so you don't need to load any extra jar file for it. Let's see a simple example of JAX-WS example in RPC style.

There are created 4 files for hello world JAX-WS example:

1. HelloWorld.java
2. HelloWorldImpl.java
3. Publisher.java
4. HelloWorldClient.java

The first 3 files are created for server side and 1 application for client side.

JAX-WS Server Code

File: HelloWorld.java

```
package com.javatpoint;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;
//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface HelloWorld{
    @WebMethod String getHelloWorldAsString(String name);
}
```

File: HelloWorldImpl.java

```
package com.javatpoint;
import javax.jws.WebService;
//Service Implementation
@WebService(endpointInterface = "com.javatpoint.HelloWorld")
```

```
public class HelloWorldImpl implements HelloWorld{
    @Override
    public String getHelloWorldAsString(String name) {
        return "Hello World JAX-WS " + name;
    }
}
```

File: Publisher.java

```
package com.javatpoint;
import javax.xml.ws.Endpoint;
//Endpoint publisher
public class HelloWorldPublisher{
    public static void main(String[] args) {
        Endpoint.publish("http://localhost:7779/ws/hello", new HelloWorldIm
pl());
    }
}
```

How to view generated WSDL

After running the publisher code, you can see the generated WSDL file by visiting the URL:

<http://localhost:7779/ws/hello?wsdl>

JAX-WS Client Code

File: HelloWorldClient.java

```
package com.javatpoint;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
public class HelloWorldClient{
    public static void main(String[] args) throws Exception {
        URL url = new URL("http://localhost:7779/ws/hello?wsdl");

        //1st argument service URI, refer to wsdl document above
        //2nd argument is service name, refer to wsdl document above
        QName qname = new QName("http://javatpoint.com/", "HelloWorld
ImplService");
        Service service = Service.create(url, qname);
        HelloWorld hello = service.getPort(HelloWorld.class);
```

```

.     System.out.println(hello.getHelloWorldAsString("javatpoint rpc"));
.   }
. }

}

```

Output:

```
Hello World JAX-WS javatpoint rpc
```

	<p>(i) Explain in detail about SOAP encoding.</p> <ul style="list-style-type: none"> For transfer between client and server in a SOAP message, we encode them in XML. <p>SOAP Encoding is an extension of the SOAP framework specification that defines how a data value should be encoded in an XML format. SOAP Data Model is defined as an adjunct in SOAP 1.2 specification.</p> <p>SOAP encoding offers the following rules to convert any data value defined in SOAP data model into XML format. Converting a data value into XML format is called serialization or encoding.</p> <p>Rule 1. A simple value node with a labeled inbound edge will be serialized into a single XML element with the edge's label as the element's name and node value as the element's text content.</p> <p>Rule 2. When serializing a node into an XML element, an "xsi:type" attribute can be added to specify the value type of this note. For more information on "xsi:type", see the other sections in this book.</p> <p>Rule 3. A compound value node with labeled outbound edges, a data structure, will be serialized into a single XML element with child elements. One outbound edge will be serialized into one child element with element's name equal to the edge's label. The order of child elements is not significant.</p> <p>Rule 4. A compound value node with non-labeled outbound edges, a data array, will be serialized into a single XML element with child elements. One outbound edge will be serialized into one child element with element's name equal to any label as long as it's the same for all child elements. The order of child elements signifies the position values of outbound edges.</p> <p>Rule 5. When serializing an array, an "enc:itemType" attribute can be added to specify the value type of its sub nodes, and an "enc:arraySize" attribute can be added to specify the number of values in the array.</p> <p>(ii) Point out the RPC representation model.</p> <h2>What Is JAX-RPC?</h2> <p>JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.</p>
13.	

In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. In this release, JAX-RPC relies on SOAP 1.1 and HTTP 1.1.

Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer. On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy.

With JAX-RPC, clients and Web services have a big advantage--the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.

- Explain** the structure of a WSDL document, its elements and their purposes with appropriate examples.
- A WSDL document defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** which are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitutes a reusable **binding**. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:
- **Types**— a container for data type definitions using some type system (such as XSD).
 - **Message**— an abstract, typed definition of the data being communicated.

- **Operation**— an abstract description of an action supported by the service.
- **Port Type**—an abstract set of operations supported by one or more endpoints.
- **Binding**— a concrete protocol and data format specification for a particular port type.
- **Port**— a single endpoint defined as a combination of a binding and a network address.
- **Service**— a collection of related endpoints.

In addition, WSDL defines a common **binding** mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions.

WSDL Document Example

The following example shows the WSDL definition of a simple service providing stock quotes. The service supports a single operation called GetLastTradePrice, which is deployed using the SOAP 1.1 protocol over HTTP. The request takes a ticker symbol of type string, and returns the price as a float. A detailed description of the elements used in this definition can be found in Section 2 (core language) and Section 3 (SOAP binding).

This example uses a fixed XML format instead of the SOAP encoding (for an example using the SOAP encoding, see [Example 4](#)).

Example 1 SOAP 1.1 Request/Response via HTTP

```

<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd1="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>
  <schema targetNamespace="http://example.com/stockquote.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="TradePriceRequest">
      <complexType>
        <all>
          <element name="tickerSymbol" type="string"/>
        </all>
      </complexType>
    </element>
    <element name="TradePrice">
      <complexType>
        <all>
          <element name="price" type="float"/>
        </all>
      </complexType>
    </element>
  </schema>
</types>

```

	<pre> </all> </complexType> </element> </schema> </types> <message name="GetLastTradePriceInput"> <part name="body" element="xsd1:TradePriceRequest"/> </message> <message name="GetLastTradePriceOutput"> <part name="body" element="xsd1:TradePrice"/> </message> <portType name="StockQuotePortType"> <operation name="GetLastTradePrice"> <input message="tns:GetLastTradePriceInput"/> <output message="tns:GetLastTradePriceOutput"/> </operation> </portType> <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType"> <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/> <operation name="GetLastTradePrice"> <soap:operation soapAction="http://example.com/GetLastTradePrice"/> <input> <soap:body use="literal"/> </input> <output> <soap:body use="literal"/> </output> </operation> </binding> <service name="StockQuoteService"> <documentation>My first service</documentation> <port name="StockQuotePort" binding="tns:StockQuoteBinding"> <soap:address location="http://example.com/stockquote"/> </port> </service> </definitions> </pre>
PAR T – C	
Q.No	<p>Questions</p> <p>Create an XML HttpRequest to retrieve data from an XML file and display the data in an HTML table. The data to be retrieved is a collection of stationary items stored in an XML file.</p> <p>1. <u>The XML Document Used</u></p> <p>INPUT: XML file called "cd_catalog.xml".</p>

Display XML Data in an HTML Table

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

Example

```
<html>
<head>
<style>
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
th, td {
    padding: 5px;
}
</style>
</head>
<body>

<table id="demo"></table>

<script>
function loadXMLDoc() {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            myFunction(this);
        }
    };
    xmlhttp.open("GET", "cd_catalog.xml", true);
    xmlhttp.send();
}
function myFunction(xml) {
    var i;
    var xmlDoc = xml.responseXML;
    var table = "<tr><th>Artist</th><th>Title</th></tr>";
    var x = xmlDoc.getElementsByTagName("CD");
    for (i = 0; i < x.length; i++) {
        table += "<tr><td>" +
        x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
        "</td><td>" +
        x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
        "</td></tr>";
    }
}
</script>
```

```

        }
        document.getElementById("demo").innerHTML = table;
    }
</script>

</body>
</html>

```

OUTPUT

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only
Dr.Hook	Sylvias Mother

Summarize Ajax Client server architecture in detail.

What is AJAX?

AJAX = Asynchronous JavaScript And XML.

AJAX is not a programming language.

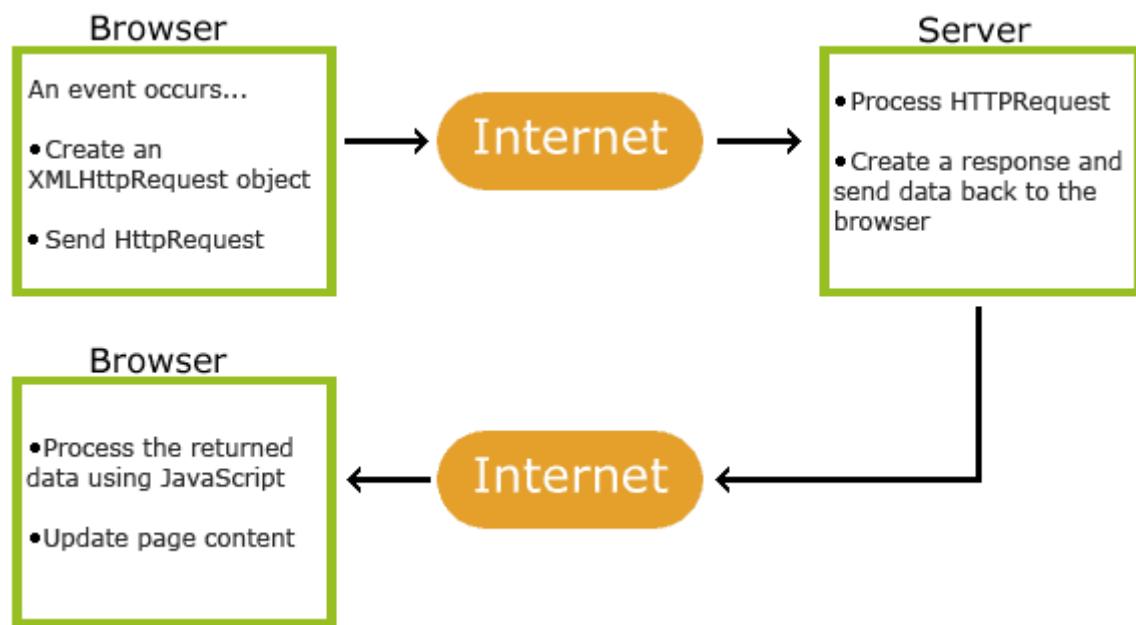
AJAX just uses a combination of:

- 2.
- A browser built-in XMLHttpRequest object (to request data from a web server)
 - JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

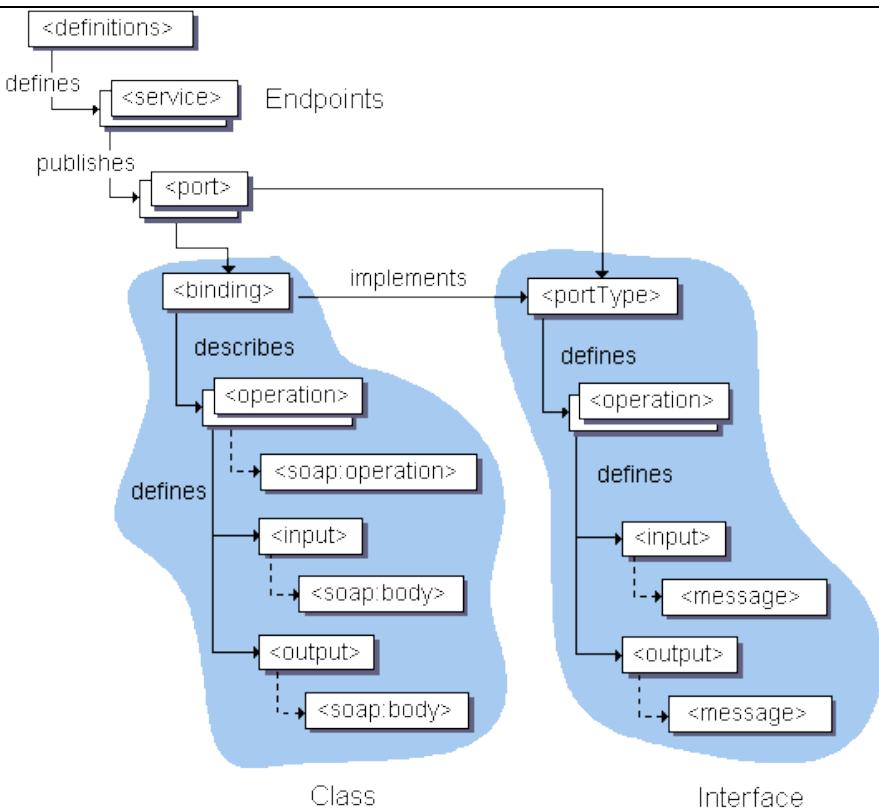
AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

How AJAX Works



1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

	<p>Give the basic structure of a WSDL and show how they are used to create, publish, test and describe web services.</p> <h3>Structure of a WSDL Document</h3> <p>Web Services Description Language (WSDL) is an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. The diagram below illustrates the elements that are present in a WSDL document, and indicates their relationships. To see an example of how this is implemented in a WSDL document, see Example of a WSDL Document.</p>
3.	



WSDL Document Elements

A WSDL document has a `definitions` element that contains the other five elements, types, message, portType, binding and service. The following sections describe the features of the generated client code.

WSDL supports the XML Schemas specification (XSD) as its type system.

definitions

Contains the definition of one or more services. JDeveloper generates the following attribute declarations for this section:

- `name` is optional.
- `targetNamespace` is the logical namespace for information about this service. WSDL documents can import other WSDL documents, and setting `targetNamespace` to a unique value ensures that the namespaces do not clash.
- `xmlns` is the default namespace of the WSDL document, and it is set to <http://schemas.xmlsoap.org/wsdl/>.
- All the WSDL elements, such as `<definitions>`, `<types>` and `<message>` reside in this namespace.

	<ul style="list-style-type: none"> • <code>xmlns:xsd</code> and <code>xmlns:soap</code> are standard namespace definitions that are used for specifying SOAP-specific information as well as data types. • <code>xmlns:tns</code> stands for this namespace. • <code>xmlns:ns1</code> is set to the value of the <code>schema targetNamespace</code>, in the <code><types></code> section.
4.	<p>Compare and contrast the additional web application architecture and AJAX Based web application architecture.</p> <p><u>Traditional Web Applications vs. Ajax Applications</u></p> <p>The following highlights the key differences between traditional web applications and Ajax-based web applications.</p> <p><i>Traditional Web Applications</i></p> <ul style="list-style-type: none"> ➤ Figure 15.1 presents the typical interactions between the client and the server in a traditional web application, such as one that uses a user registration form. ➤ First, the user fills in the form's fields, then submits the form (Fig. 15.1, <i>Step 1</i>). The browser generates a request to the server, which receives the request and processes it (<i>Step 2</i>). ➤ The server generates and sends a response containing the exact page that the browser will render (<i>Step 3</i>), which causes the browser to load the new page (<i>Step 4</i>) and temporarily makes the browser window blank. Note that the client <i>waits</i> for the server to respond and <i>reloads the entire page</i> with the data from the response (<i>Step 4</i>). ➤ While such a synchronous request is being processed on the server, the user cannot interact with the client web page.

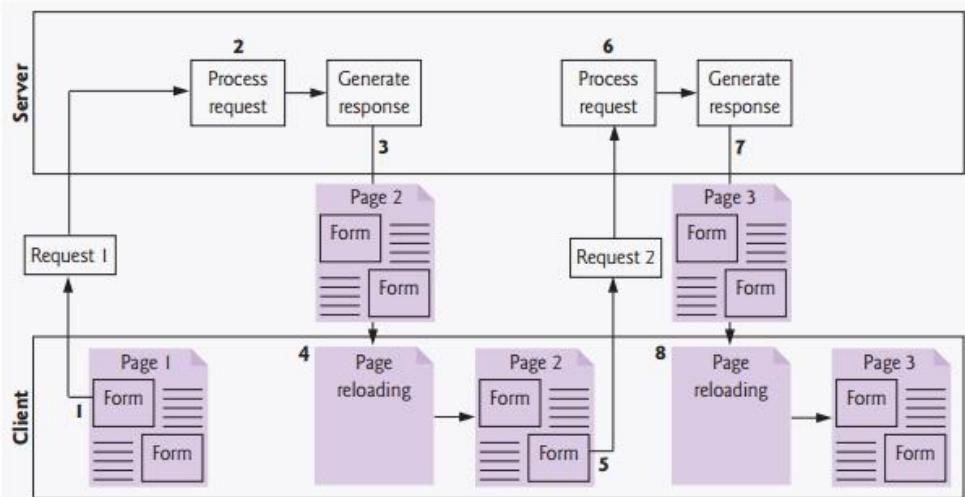


Fig. 15.1 | Classic web application reloading the page for every user interaction.

- Frequent long periods of waiting, due perhaps to Internet congestion, have led some users to refer to the World Wide Web as the “World Wide Wait.”
- If the user interacts with and submits another form, the process begins again (*Steps 5–8*).

This model was originally designed for a web of hypertext documents—what some people call the “brochure web.”

As the web evolved into a full-scale applications platform, the model shown in Fig. 15.1 yielded “choppy” application performance.

Every full-page refresh required users to re-establish their understanding of the full-page contents.

Users began to demand a model that would yield the responsive feel of desktop applications.

Ajax Web Applications

- Ajax applications add a layer between the client and the server to manage communication between the two (Fig. 15.2). When the user interacts with the page, the client creates an XMLHttpRequest object to manage a request (*Step 1*).
- The XMLHttpRequest object sends the request to the server (*Step 2*) and awaits the response.
- The requests are **asynchronous**, so the user can continue interacting with the application on the client-side while the server processes the

earlier request concurrently. Other user interactions could result in additional requests to the server (*Steps 3 and 4*).

- Once the server responds to the original request (*Step 5*), the XMLHttpRequest object that issued the request calls a client-side function to process the data returned by the server.
- This function—known as a **callback function**—uses **partial page updates** (*Step 6*) to display the data in the existing web page *without re-loading the entire page*. At the same time, the server may be responding to the second request (*Step 7*) and the client-side may be starting to do another partial page update (*Step 8*).
- The callback function updates only a designated part of the page.
- Such partial page updates help make web applications more responsive, making them feel more like desktop applications.
- The web application does not load a new page while the user interacts with it.

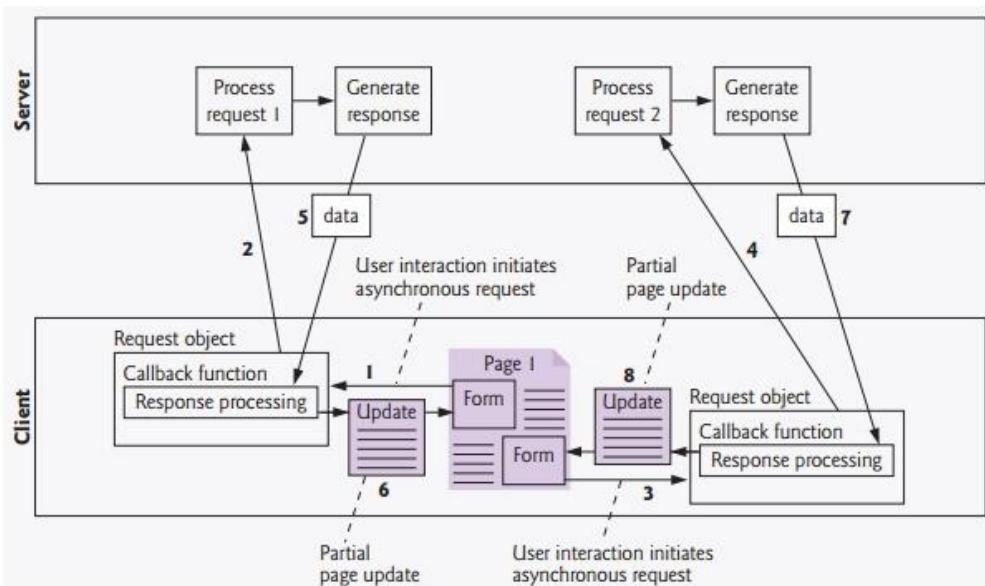


Fig. 15.2 | Ajax-enabled web application interacting with the server asynchronously.

UNIT I

WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0

Web Essentials: Clients, Servers and Communication – The Internet – Basic Internet protocols – World wide web – HTTP Request Message – HTTP Response Message – Web Clients – Web Servers – HTML5 – Tables – Lists – Image – HTML5 control elements – Semantic elements – Drag and Drop – Audio – Video controls - CSS3 – Inline, embedded and external style sheets – Rule cascading – Inheritance – Backgrounds – Border Images – Colors – Shadows – Text – Transformations – Transitions – Animations.

TEXT BOOK:

1.Deitel and Deitel and Neito,"Internet and World wide web-How to Program",Prentice Hall,5th Edition,2011

REFERENCES:

- 1.Stephen Wynkoop and John Burke —Running a Perfect Website, QUE, 2nd Edition,1999.
- 2.Chris Bates, Web Programming – Building Intranet Applications, 3rd Edition, Wiley Publications, 2009.
- 3.Jeffrey C and Jackson, —Web Technologies A Computer Science Perspective, Pearson Education, 2011.
- 4.Gopalan N.P. and Akilandeswari J., —Web Technology, Prentice Hall of India, 2011.
- 5.UttamK.Roy, —Web Technologies, Oxford University Press, 2011.

Faculty Incharge

HOD/CSE

1. WEB ESSENTIALS:CLIENT,SERVERS AND COMMUNICATON

1.1 THE INTERNET:

- The Internet traces its roots to a project of the U.S. Department of Defense's then named Advanced Research Projects Agency, or ARPA. The ARPANET project was intended to support DoD research on computer networking.
- As this project began in the late 1960s, there had been only a few small experimental networks providing communication between geographically dispersed computers from different manufacturers running different operating systems.
- The purpose of ARPANET was to create a larger such network, both in order to electronically connect DoD-sponsored researchers and in order to experiment with and develop tools for heterogeneous computer networking.
- The ARPANET computer network was launched in 1969 and by year's end consisted of four computers at four sites running four different operating systems.
- **For example**, e-mail was available on ARPANET beginning in 1972, and it soon became an extremely popular application for those who had ARPANET access.
- The regional U.S. networks were often cooperative efforts between universities.
- As one example, SURAnet (Southeastern University Research Association Network) was organized by the University of Maryland beginning in 1982 and eventually included essentially all of the major universities and research institutions in the southeastern United States.
- Another of these networks, CSNET (Computer Science Network), was partially funded by the U.S. National Science Foundation (NSF) to aid scientists at universities without ARPANET access, laying the groundwork for future network developments that we'll say more about in a moment.
- Several of the most widely used Internet protocols—including the File Transfer Protocol (FTP) and Simple Mail Transfer Protocol (SMTP), which underlie many of the Internet's file transfer and e-mail operations, respectively—were initially developed under ARPANET.
- ARPANET switched from using an earlier protocol to TCP/IP during 1982. At around the same time, an ARPA Internet was created, allowing computers on some outside networks such as CSNET to communicate via TCP/IP with computers on the ARPANET.
- One of the primary goals of this network was to connect the NSF's new regional supercomputing centers. But it was also decided that regional networks should be able to connect to NSFNET, so that the NSFNET would provide a backbone through which other networks could interconnect synchronously.
- NSFNET quickly supplanted ARPANET, which was officially decommissioned in 1990. At this point, NSFNET was at the center of the Internet, that is, the collection of computer networks connected via the public backbone and communicating across networks using TCP/IP.
- One of the arguments for allowing commercial traffic was economic: commercial traffic would increase network usage, leading to reduced unit costs through economies of scale.
- This in turn would provide a less expensive network for research and educational purposes.

1.2 BASIC INTERNET PROTOCOLS:

A computer communication protocol is a detailed specification of how communication between two computers will be carried out in order to serve some purpose.

1.2.1 TCP/IP

- The reason that they are often treated as one is that the bulk of the services we associate with the Internet—e-mail, Web browsing, file downloads, accessing remote databases—are built on top of both the TCP and IP protocols.
- But in reality, only one of these protocols—IP, the Internet Protocol—is fundamental to the definition of the Internet.
- A key element of IP is the IP address, which is simply a 32-bit number.
- IP addresses are normally written as a sequence of four decimal numbers separated by periods (called “dots”), as in 192.0.34.166.
- Each decimal number represents one byte of the IP address. The function of IP software is to transfer data from one computer (the source) to another computer (the destination).
- The IP software running on the source creates a packet, which is a sequence of bits representing the data to be transferred along with the source and destination IP addresses and some other header information, such as the length of the data.
- If the destination computer is on the same local network as the source, then the IP software will send the packet to the destination directly via this network.
- If the destination is on another network, the IP software will send the packet to a gateway, which is a device that is connected to the source computer’s network as well as to at least one other network.
- The gateway will select a computer on one of the other networks to which it is attached and send the packet on to that computer.
- This process will continue, with the packet going through perhaps a dozen or more hops, until the packet reaches the destination computer.
- IP software on that computer will receive the packet and pass its data up to an application that is waiting for the data.
- The sequence of computers that a packet travels through from source to destination is known as its route. How does each computer choose the next computer in the route for a packet?
- A separate protocol (the current standard is BGP-4, the Border Gateway Protocol) is used to pass network connectivity information between gateways so that each can choose a good next hop for each packet it receives.
- IP software also adds some error detection information (a checksum) to each packet it creates, so that if a packet is corrupted during transmission, this can usually be detected by the recipient. The IP standard calls for IP software to simply discard any corrupted packets.
- TCP, the Transmission Control Protocol, is a higher-level protocol that extends IP to provide additional functionality, including reliable communication based on the concept of a connection.
- A and B can both send messages to one another at the same time; this is known as full duplex communication. When A and B are both done sending messages to one another (or at least done for the time being), a similar set of three messages is used to close the connection.
- Once a connection has been established, TCP provides reliable data transmission by demanding an acknowledgment for each packet it sends via IP.

- Essentially, the software sets a timer after sending each packet.
- The TCP software on the receiving side sends a packet containing an acknowledgment for every TCP-based packet it receives that passes the checksum test.
- If the TCP software sending a packet does not receive an acknowledgment packet before its timer expires, then it resends the packet and restarts the timer.
- Another important feature that TCP adds to IP is the concept of a port.
- The port concept allows TCP to be used to communicate with many different applications on a machine.

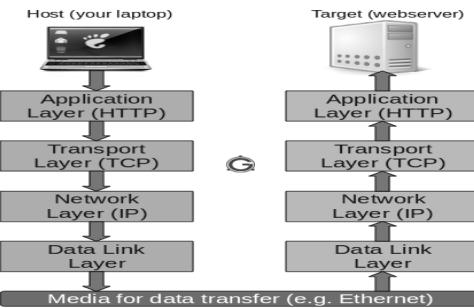


Fig 1.1:Simplified view of communication using TCP/IP

1.2.2 UDP, DNS, and Domain Names

- UDP (User Datagram Protocol) is an alternative protocol to TCP that also builds on IP.
- The main feature that UDP adds to IP is the port concept that we have just seen in TCP.
- However, it does not provide the two-way connection or guaranteed delivery of TCP. Its advantage over TCP is speed for simple tasks.
- One Internet application that is often run using UDP rather than TCP is the Domain Name Service (DNS). DNS provides a mechanism for mapping back and forth between IP addresses and host names.
- IP address—it uses the UDP software running on its system to send a UDP message to one of these DNS servers, requesting the IP address.
- If all goes well, this server will then send back a UDP message containing the IP address. Recall that it took three messages just to get a TCP connection set up, so the UDP approach is much more efficient for sporadic DNS queries. (UDP is sometimes referred to as a lightweight communication protocol and TCP as a heavyweight protocol, at least in comparison with UDP.)
- In general, the terms lightweight and heavyweight in computer science are used to describe alternative software solutions to some problem, with the lightweight solution having less functionality but also less overhead.)
- Internet host names consist of a sequence of labels separated by dots. The final label in a host name is a top-level domain. There are two standard types of top-level domain: generic (such as .com, .edu, .org, and .biz) and country-code (such as .de, .il, and .mx).
- Each top-level domain is divided into subdomains (second-level domains), which may in turn be further divided, and so on. The assignment of second-level domains within each top-level domain is performed (for a fee) by a registry operator selected by ICANN.

- Such a subdomain, consisting of a local host name followed by a domain name (typically consisting of at least two labels) is sometimes called a fully qualified domain name for the computer.
- For example, www.example.org is a fully qualified domain name for a host with local name www that belongs to the example second-level domain of the org top-level domain.
- Typical usage of nslookup is illustrated by the following (user input is italicized):

```
C:\>nslookup www.example.org
Server: slave9.dns.stargate.net
Address: 209.166.161.121
```

```
Name: www.example.org
Address: 192.0.34.166
```

```
C:\>nslookup 192.0.34.166
Server: slave9.dns.stargate.net
Address: 209.166.161.121
```

```
Name: www.example.com
Address: 192.0.34.166
```

- The first two lines following the command line identify the qualified name and IP address of the DNS server that is providing the domain name information that follows. Also notice that a single IP address can be associated with multiple domain names. In this example, both www.example.org and www.example.com are associated with the IP address 192.0.34.166.
- A lookup that specifies an IP address, such as the second lookup in the example, is sometimes referred to as a reverse lookup. As shown, even if multiple qualified names are associated with an IP address, only one of the names will be returned by a reverse lookup. This is known as the canonical name for the host; all other names are considered aliases.
- The reverse lookup in the example indicates that www. example. com is the canonical name for the host with IP address 192.0.34.166.

1.2.3 Higher-Level Protocols

In the cases of both TCP and a phone call, different protocols can be used to communicate once a connection has been established. Similarly, a variety of higher-level protocols are used to communicate once a TCP connection has been established.

- SMTP supports transfer of e-mail between different e-mail servers, while FTP is used for transferring files between machines. Another higher-level TCP protocol, Telnet, is used to execute commands typed into one computer on a remote computer.
- Telnet can also be used to communicate directly (via keyboard entries) with some TCP-based applications. As described earlier, which protocol will be used to communicate over a TCP connection is normally determined by the port number used to establish the connection.
- The primary TCP-based protocol used for communication between web servers and browsers is called the Hypertext Transport Protocol (HTTP).

1.3 THE WORLD WIDE WEB

- The Usenet newsgroup service began in 1979 and provided a means of “posting” information.
- Large files were (and still are) often shared by running an FTP server application that allowed any user to transfer the files from their origin machine to the user’s machine.
- The first Internet chat software in widespread use, Internet Relay Chat (IRC), provided both public and private chat facilities.
- Some of the more popular information management technologies in the early 1990s were Gopher information servers, which provided a simple hierarchical view of documents; the Wide Area Information System (WAIS) system for indexing and retrieving information; and the ARCHIE tool for searching online information archives accessible via FTP.
- The server and client applications communicate over the Internet by following a communication protocol built on top of TCP/IP.
- The protocol used by the Web, as just noted, is the Hypertext Transport Protocol, HTTP. As we will learn in the next section, this is a rather generic protocol that for the most part supports a client requesting a document from a server and the server returning the requested document.
- This generic nature of HTTP gives it the advantage of somewhat more flexibility than is present in the protocols used by WAIS and Gopher.
- Most web pages are written using the Hypertext Markup Language, HTML, which along with HTTP is a fundamental web technology. HTML pages can contain the familiar web links (technically called hyperlinks) to other documents on the Web.
- In addition to hyperlinks, modern versions of HTML also provide extensive page layout facilities, including support for inline graphics, which (as you might guess) has added significantly to the commercial appeal of the Web.

1.3.1 Hypertext Transport Protocol

- HTTP is a form of communication protocol, in particular a detailed specification of how web clients and servers should communicate.
- The basic structure of HTTP communication follows what is known as a request-response model. Specifically, the protocol dictates that an HTTP interaction is initiated by a client sending a request message to the server; the server is then expected to generate a response message.
- The format of the request and response messages is dictated by HTTP.
- HTTP does not dictate the network protocol to be used to send these messages, but does expect that the request and response are both sent within a TCP-style connection between the client and the server.
- So most HTTP implementations send these messages using TCP.
- When I pressed the Enter key after typing this address, the browser created a message conforming to the HTTP protocol, used DNS to obtain an IP address for www.example.org, created a TCP connection with the machine at the IP address obtained, sent the HTTP message over this TCP connection, and received back a message containing the information that is shown displayed in the client area of the browser.
- A nice feature of HTTP is that these request and response messages often consist entirely of plain text in a fairly readable form.

- An HTTP request message consists of a start line followed by a message header and optionally a message body.
- The request consists of the three lines beginning with the GET and ending with a blank line, and user input is again italicized):
- The response message in this case begins with the line which is known as the status line of the response, and continues to the end of the example.

HTTP/1.1 200 OK

- The portion of the response between the status line and the first blank line following it is the header of the response.

1.4 HTTP REQUEST MESSAGE

1.4.1 Overall Structure

- Every HTTP request message has the same basic structure:
 - Start line
 - Header field(s) (one or more)
 - Blank line
 - Message body (optional)
- Every start line consists of three parts, with a single space used to separate adjacent parts:
 - Request method
 - Request-URI portion of web address
 - HTTP version

1.4.2 HTTP Version

The initial version of HTTP was referred to as HTTP/0.9, and the first Internet RFC regarding .HTTP described HTTP/1.0. In 1997, HTTP/1.1 was formally defined, and is currently an Internet Draft Standard [RFC-2616].

1.4.3 Request-URI

- The second part of the start line is known as the Request-URI. The concatenation of the string http://, the value of the Host header field (www.example.org, in this example), and the Request-URI (/ in this example) forms a string known as a Uniform Resource Identifier (URI).
- A URI is an identifier that is intended to be associated with a particular resource (such as a web page or graphics image) on the World Wide Web. Every URI consists of two parts: the scheme, which appears before the colon (:), and another part that depends on the scheme.
- Web addresses, for the most part, use the http scheme (the scheme name in URIs is case insensitive, but is generally written in lowercase letters). In this scheme, the URI represents the location of a resource on the Web. A URI of this type is said to be a Uniform Resource Locator (URL).

Scheme Name	Example URL	Type of Resource
ftp	ftp://ftp.example.org/pub/afile.txt	File located on FTP server
telnet	telnet://host.example.org/	Telnet server
mailto	mailto:someone@example.org	Mailbox
https	https://secure.example.org/sec.txt	Resource on web server supporting encrypted communication
file	file:///C:/temp/localFile.txt	File accessible from machine processing this URL

TABLE 1.1 Some Non-http URL Schemes

- In addition to the URL type of URI, there is one other type, called a Uniform Resource Name (URN).
- The URI for a URN always consists of three colon-separated parts, as illustrated here.
- The first part is the scheme name, which is always urn for a URN-type URI.
- The second part is the namespace identifier, which in this example is ISBN. Other currently registered URN namespace identifiers along with pointers to documentation for each are listed at [IANA- URNS].
- The third part is the namespace-specific string. The exact format and meaning of this string varies with the namespace.

1.4.4 Request Method

- The HTTP/1.1 standard defines a CONNECT method, which can be used to create certain types of secure connections.
- The primary HTTP method is GET. This is the method used when you type a URL into the Location bar of your browser.
- It is also the method that is used by default when you click on a link in a document displayed in your browser and when the browser downloads images for display within an HTML document.
- The POST method is typically used to send information collected from a form displayed within a browser, such as an order-entry form, back to the web server.

Method	Requests server to ...
GET	return the resource specified by the Request-URI as the body of a response message.
POST	pass the body of this request message on as data to be processed by the resource specified by the Request-URI.
HEAD	return the same HTTP header fields that would be returned if a GET method were used, but not return the message body that would be returned to a GET (this provides information about a resource without the communication overhead of transmitting the body of the response, which may be quite large).
OPTIONS	return (in Allow header field) a list of HTTP methods that may be used to access the resource specified by the Request-URI.
PUT	store the body of this message on the server and assign the specified Request-URI to the data stored so that future GET request messages containing this Request-URI will receive this data in their response messages.
DELETE	respond to future HTTP request messages that contain the specified Request-URI with a response indicating that there is no resource associated with this Request-URI.
TRACE	return a copy of the complete HTTP request message, including start line, header fields, and body, received by the server. Used primarily for test purposes.

Table 1.2 Standard HTTP/1.1 Methods

1.4.5 Header Fields and MIME Types

- The Host header field is required in every HTTP/1.1 request message.
- HTTP/1.1 also defines a number of other header fields, several of which are commonly used by modern browsers. Each header field begins with a field name, such as Host, followed by a colon and then a field value.
- The following slightly modified example of an actual HTTP request sent by a browser consists of a start line, 10 header fields, and a short message body:

host: www.example.org:56789

*user-agent: Mozilla/S.O (Windows; U; Windows NT 5.1; en-US; rv:1.4)
Gecko/20030624
accept: text/xml,application/xml,application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,
image/gif;q=0.2,*/*;q=0.1
accept-language: en-us,en;q=0.5
accept-encoding: gzip,deflate
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
connection: keep-alive
keep-alive: 300
content-type: application/x-www-form-urlencoded
content-length: 13*

- First, header names are not case sensitive, although I will throughout this text refer to header field names following the capitalization used by the HTTP/1.1 reference [RFC-2616]. So, while the browser used “host” to name the first header Field.
- Second, a header field value may wrap onto several lines by preceding each continuation line with one or more spaces or tabs, as shown for the User-Agent and Accept fields of the preceding example. This also means that a header field name must begin at the first character of a line, with no preceding white space.
- A third common feature is the use of so-called MIME types in several header field values. MIME is an acronym standing for Multipurpose Internet Mail Extensions, and refers to a standard that can be used to pass a variety of types of information, including graphics and applications, through e-mail as well as through other Internet message protocols.

Top-level Content Type	Document Content
application	Data that does not fit within another content type and that is intended to be processed by application software, or that is itself an executable binary.
audio	Audio data. Subtype defines audio format.
image	Image data, typically static. Subtype defines image format. Requires appropriate software and hardware in order to be displayed.
message	Another document that represents a MIME-style message. For example, following an HTTP TRACE request message to a server, the server sends a response with a body that is a copy of the HTTP request. The value of the Content-Type header field in the response is message/http.
model	Structured data, generally numeric, representing physical or behavioral models.
multipart	Multiple entities, each with its own header and body.
text	Displayable as text. That is, a human can read this document without the need for special software, although it may be easier to read with the assistance of other software.
video	Animated images, possibly with synchronized sound.

Table 1.3 Standard Top-level MIME Content Types

MIME Type	Description
text/html	HTML document
image/gif	Image represented using Graphics Interchange Format (GIF)
image/jpeg	Image represented using Joint Picture Expert Group (JPEG) format
text/plain	Human-readable text with no embedded formatting information
application/octet-stream	Arbitrary binary data (may be executable)
application/x-www-form-urlencoded	Data sent from a web form to a web server for processing

Table 1.4 Some Common MIME Content Types

1.5 HTTP RESPONSE MESSAGE

- An HTTP response message consists of a status line, header fields, and the body of the response, in the following format:

Status line

Header field(s) (one or more)

Blank line

Message body (optional)

1.5.1 Response Status Line

The example status line shown earlier was

HTTP/1.1 200 OK

Field Name	Use
Host	Specify <i>authority</i> portion of URL (host plus port number; see Section 1.6.2). Used to support <i>virtual hosting</i> (running separate web servers for multiple fully qualified domain names sharing a single IP address).
User-Agent	A string identifying the browser or other software that is sending the request.
Accept	MIME types of documents that are acceptable as the body of the response, possibly with indication of preference ranking. If the server can return a document according to one of several formats, it should use a format that has the highest possible preference rating in this header.
Accept-Language	Specifies preferred language(s) for the response body. A server may have several translations of a document, and among these should return the one that has the highest preference rating in this header field. For complete information on registered language tags, see [RFC-3066] and [ISO-639-2].
Accept-Encoding	Specifies preferred encoding(s) for the response body. For example, if a server wishes to send a compressed document (to reduce transmission time), it may only use one of the types of compression specified in this header field.
Accept-Charset	Allows the client to express preferences to a server that can return a document using various character sets (see Section 1.5.4).
Connection	Indicates whether or not the client would like the TCP connection kept open after the response is sent. Typical values are <i>keep-alive</i> if connection should be kept open (the default behavior for servers/clients compatible with HTTP/1.1), and <i>close</i> if not.
Keep-Alive	Number of seconds TCP connection should be kept open.
Content-Type	The MIME type of the document contained in the message body, if one is present. If this field is present in a request message, it normally has the value shown in the example, <i>application/x-www-form-urlencoded</i> .
Content-Length	Number of bytes of data in the message body, if one is present.
Referer	The URI of the resource from which the browser obtained the Request-URI value for this HTTP request. For example, if the user clicks on a hyperlink in a web page, causing an HTTP request to be sent to a server, the URI of the web page containing the hyperlink will be sent in the Referer field of the request. This field is not present if the HTTP request was generated by the user entering a URI in the browser's Location bar.

Table 1.5 Some Common HTTP/1.1 Request Header Fields

- The status line consists of three fields: the HTTP version used by the server software when formatting the response; a numeric status code indicating the type of response; and a text string (the reason phrase) that presents the information represented by the numeric status code in human-readable form.
- The first digit represents the general class of status code. The five classes of HTTP/1.1 status codes are given in Table 1.6. The last two digits of a status code define the specific status within the specified class.

Digit	Class	Standard Use
1	Informational	Provides information to client before request processing has been completed.
2	Success	Request has been successfully processed.
3	Redirection	Client needs to use a different resource to fulfill request.
4	Client Error	Client's request is not valid.
5	Server Error	An error occurred during server processing of a valid client request.

Table 1.6 HTTP/1.1 Status Code Classes (First Digit of Status Code)

1.5.2 Response Header Fields

- Some of the header fields used in HTTP request messages, including Connection, Content-Type, and Content-Length, are also valid in response messages. The Content-Type of a response can be any one of the MIME type values specified by the Accept header field of the corresponding request. Some other common response header fields are shown in Table 1.8.

Status Code	Recommended Reason Phrase	Usual Meaning
200	OK	Request processed normally.
301	Moved Permanently	URI for the requested resource has changed. All future requests should be made to URI contained in the Location header field of the response. Most browsers will automatically send a second request to the new URI and display the second response.
307	Temporary Redirect	URI for the requested resource has changed at least temporarily. This request should be fulfilled by making a second request to URI contained in the Location header field of the response. Most browsers will automatically send a second request to the new URI and display the second response.
401	Unauthorized	The resource is password protected, and the user has not yet supplied a valid password.
403	Forbidden	The resource is present on the server but is read protected (often an error on the part of the server administrator, but may be intentional).
404	Not Found	No resource corresponding to the given Request-URI was found at this server.
500	Internal Server Error	Server software detected an internal failure.

Table 1.7 Some Common HTTP/1.1 Status Codes

Field Name	Use
Date	Time at which response was generated. Used for cache control (see Section 1.5.3). This field must be supplied by the server.
Server	Information identifying the server software generating this response.
Last-Modified	Time at which the resource returned by this request was last modified. Can be used to determine whether cached copy of a resource is valid or not (see Section 1.5.3).
Expires	Time after which the client should check with the server before retrieving the returned resource from the client's cache (see Section 1.5.3).
ETag	A hash code of the resource returned. If the resource remains unchanged on subsequent requests, then the ETag value will also remain unchanged; otherwise, the ETag value will change. Used for cache control (see Section 1.5.3).
Accept-Ranges	Clients can request that only a portion (<i>range</i>) of a resource be returned by using the Range header field. This might be used if the resource is, say, a large PDF file and only a single page is currently needed. Accept-Ranges specifies the units that may be used by the client in a range request, or none if range requests are not accepted by this server for this resource.
Location	Used in responses with redirect status code to specify new URI for the requested resource.

Table 1.8 Some Common HTTP/1.1 Response Header Fields

1.5.3 Cache Control

- In computer systems, a cache is a repository for copies of information that originates elsewhere. A copy of information is placed in a cache in order to improve system performance.
- Most web browsers automatically cache on the client machine many of the resources that they request from servers via HTTP.
- However, there is a key drawback to using a cache: information in a cache can become invalid.
- One approach to guaranteeing that a cached copy of a resource is valid is for the client to ask the server whether or not the client's copy is valid. This can be done with relatively little communication by sending an HTTP request for the resource using the HEAD method, which returns only the status line and header portion of the response.
- If the response message contains a Last-Modified time, and this time precedes the value of the Date header field returned with the cached resource, then the cached copy is still valid and can be used. Otherwise, the cached copy is invalid and the browser should send a normal GET request for the resource.

1.5.4 Character Sets

- Character set defines the mapping between these integers, or code points, and characters.
- A character encoding is a bit string that must be decoded into a code-point integer that is then mapped to a character according to the definition provided by some character set. A character encoding often represents characters using variable-length bit strings, with common characters represented using shorter strings and less-common characters using longer strings.
- The Accept-Charset header field is used by a client to tell a server the character sets and character encodings that it will accept as well as its preferred character sets or encodings, if more than one is available for the requested document. In our earlier example, the header field

accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

said that the client would prefer to receive documents using the ISO-8859-1 character set or the UTF-8 encoding of the characters in Unicode, but that it would also accept any other valid Internet character set/encoding.

- A web server can inform a client about the character set/encoding used in a returned document by adding a charset parameter to the value of the Content-Type header field.
- For example, the following Content-Type header field in an HTTP response would indicate that the body of the message is an HTML document written using the UTF-8 character encoding:Content-Type: text/html; charset=UTF-8
- The US-ASCII character set is a subset of both the ISO-8859-1 character set and the UTF-8 character encodings, so the charset parameter is set to one of these two values for many US-ASCII documents in order to ensure international compatibility.

1.6 WEB CLIENTS

- A web client is software that accesses a web server by sending an HTTP request message and processing the resulting HTTP response.
- In general, any web client that is designed to directly support user access to web servers is known as a user agent. Furthermore, some web clients are not designed to

be used directly by humans at all. For example, software robots are often used to automatically crawl the Web and download information for use by search engines

1.6.1 Basic Browser Functions

- The window of a typical modern browser is split into several rectangular regions, most of which are known as bars. Figure 1.4 shows five standard regions in a Mozilla 1.4 window.
- The primary region is the client area, which displays a document. For many documents, the title bar displays a title assigned by the document author to the document currently displayed within the client area. The title bar also displays the browser name as well as standard window-management controls. The menu bar contains a set of dropdown menus, much like most other applications that incorporate a graphical user interface (GUI).
- The browser's Navigation toolbar contains standard push-button controls that allow the user to return to a previously viewed web page (Back), reverse the effect of pressing Back (Forward), ask the server for an updated version of the page currently viewed (Reload), halt page downloading currently in progress (Stop), and print the client area of the window (Print). Clicking the small down-arrow to the right of some buttons produces a menu allowing users to override the default behavior of the associated button.
- The Navigation toolbar also contains a text box, known as the Location bar, where a user can enter a URL and press the Enter key in order to request the browser to display the document located at the specified URL.
- Clicking the Search button instead of pressing Enter causes the information entered in the text box to be sent to a search engine. Clicking the down-arrow at the right side of the Location bar produces a dropdown menu of recently visited URLs that can be visited again with a single click. Finally, the status bar displays messages and icons related to the status of the browser.
- A primary task of any browser is to make HTTP requests on behalf of the browser user.
- If a user types an http-scheme URL in Mozilla's Location bar, for example, the browser must perform a number of tasks:
 - Reformat the URL entered as a valid HTTP request message.
 - If the server is specified using a host name (rather than an IP address), use DNS to convert this name to the appropriate IP address.
 - Establish a TCP connection using the IP address of the specified web server.
 - Send the HTTP request over the TCP connection and wait for the server's response.
 - Display the document contained in the response. If the document is not a plain-text document but instead is written in a language such as HTML, this involves rendering the document: positioning text and graphics appropriately within the browser window, creating table borders, using appropriate fonts and colors, etc.

1.6.2 URLs

- An http-scheme URL consists of a number of pieces. In order to show the main possibilities, let's consider the following example URL:

http://www.example.org:56789/a/b/c.txt?t=win&s=chess#para5

- The portion of an http URL following the :// string and before the next slash (/) (or through the completion of the URL, if there is no trailing slash) is known as the authority of the URL.
- It consists of either a fully qualified domain name (or other name that can be resolved to an IP address, such as an unqualified name of a machine on the local network) or an IP.

Status Message	Meaning
Resolving host www.example.org ...	Requested IP address from DNS; waiting for response.
Connecting to www.example.org ...	Creating TCP connection to server.
Waiting for www.example.org ...	Sent HTTP request to server; waiting for HTTP response.
Transferring data from www.example.org ...	HTTP response has begun, but has not completed.
Done	HTTP response has been received, although further processing may be needed before the document will be displayed.

Table 1.9 Some Mozilla Status Messages

- if the port number is omitted, then a TCP connection to port 80 is implied. In this example, the authority is www.example.org:56789 and consists of the fully qualified domain name www.example.org followed by the port number 56789.
- The portion from the slash following the authority through the question mark (?) (or through the end of the URL, if there is no question mark) is called the path of the URL.
- The leading slash is part of the path, but the question mark is not. So the path in the example

URL just given is /a/b/c.txt.

- Following the path there may be a question mark followed by information up to a number sign (#). The information between but not including the question mark and number sign is the query portion of the URL, and in general a string of the form shown is known as a query string.
- The query portion of the example URL is t=win&s=chess. Originally, the query portion of a URL was intended to pass search terms to a web server.
- So in this example, it might be that the user is seeking a resource with a title containing the string “win” that is related to the subject “chess.”
- A browser forms the Request-URI portion of an HTTP request from a URL by concatenating the path and query portions of the URL with an intervening question mark. Thus, the Request-URI for the example URL would be

/a/b/c.txt?t=win&s=chess

- The final optional part of an http-scheme URL—the portion following but not including the number sign—is known as the fragment of the URL, and the string contained in the fragment is known as a fragment identifier. Fragment identifiers are used by browsers to scroll HTML documents; details are given in the next chapter, which covers HTML.

1.6.3 User-Controllable Features

- Graphical browsers also provide many user-controllable features, including:
 - **Save:** Most documents can be saved by the user to the client machine’s file system. If the document is an HTML page that contains other documents, such as images, then the browser will attempt to save all of these documents

locally so that the entire page can be displayed from the local file system. A user saves a document in Mozilla under the **File|Save Page As** menu.

- **Find in page:** Standard documents (text and HTML) can be searched with a function that is similar to that provided by most word processors. In Mozilla, the find function is accessed under the **Edit|Find in This Page** menu. (Mozilla also provides a “find as you type” feature under Edit that is similar to the incremental search in Emacs, for users familiar with that paradigm.)
- **Automatic form filling:** The browser can “remember” information entered on certain forms, such as billing address, phone numbers, etc. When another form is visited at a later date, the browser can automatically fill in previously saved data. The **Edit|Save Form Info** and **Edit|Fill in Form** menu options can be used to save and retrieve form information in Mozilla. The **Tools|Form Manager** menu can be used to manage saved form information.
- **Preferences:** Users can customize browser functionality in a wide variety of ways. In Mozilla, a window presenting preference options is obtained by selecting **Edit| Preferences** (Figure 1.5). The Appearance, Navigator, and Advanced categories (left subwindow) and their subcategories are used to customize Mozilla. Some preference settings directly related to the HTTP topics covered earlier are:
 - Accept-Language: The non-* values sent by the browser for this HTTP request header field can be set under the **Navigator|Languages** category, **Languages for Web Pages** box.
 - Default character set/encoding: The character set/encoding to be assumed for documents that do not specify one is also set under **Navigator|Languages** in the **Character Coding** box.
 - Cache properties: The amount of local storage allocated to the cache and the conditions controlling when a cached file will be validated are set under **Advanced| Cache** in the **Set Cache Options** box.
 - HTTP settings: The version of HTTP used and whether or not the client will keep connections alive is set under **Advanced|HTTP Networking** in the **Direct Connection Options** box.
 - Style definition: The user can define certain aspects affecting how the browser renders HTML pages, such as font sizes, background and foreground colors, etc. In Mozilla, the font size can be modified using **View|Text Zoom**. If a page offers alternative styles, they can be selected using the **View|Use Style** menu .
 - Document meta-information: Interested users can view information about the displayed document, such as the document’s MIME type, character encoding, size, and, if the document was written using HTML, the raw HTML source from which the rendering in the client area was produced. In Mozilla, **View|Page Source** is used to view raw HTML, and **View|Page Info** to view other so-called meta-information, that is, information about the document rather than information contained in the document itself.
 - Themes: The look of one or more of the browser bars, particularly the navigation bar, can be modified by applying a certain theme (sometimes called a “skin”). In Mozilla, the browser scheme can be modified using **View|Apply Theme**. Additional themes can be obtained from **View|Apply Theme|Get New Themes**.

- **History:** The browser will automatically maintain a list of all pages visited within the last several days. Users can use the history list to easily return to any recently visited page. In Mozilla, the history list can be reached by selecting **Go|History**.
- **Bookmarks** (“favorites” in Internet Explorer): Users can explicitly bookmark a web page, that is, save the URL for that page for an indefinite length of time. At any later time, the browser’s bookmark facility can be used to easily return to any bookmarked page. Options under the **Bookmarks** menu in Mozilla allow users to bookmark a page, return to a bookmarked page, and edit the bookmark list.

1.6.4 Additional Functionality

- In addition to the facilities for end users described in the preceding subsection, browsers perform a number of other functions, including:
 - **Automatic URL completion:** If the user has entered a URL in the Location bar and begins to type it again (within the next several days), the URL will be completed automatically by the browser.
 - **Script execution:** In addition to displaying documents, browsers can run programs (scripts). These programs can perform a variety of tasks, from validating data entered on a form before sending it to a web server to creating various dynamic effects on web pages, such as drop-down menus.
 - **Event handling:** When the user performs an action, such as clicking on a link or a button in a web page, the browser treats this as the occurrence of an event. Browsers recognize a number of different types of events, including mouse button clicks, mouse movement, and even events not directly under user control such as the completion of the browser’s rendering of a document. A browser can perform a variety of actions in response to an event—loading a document from a URL, clearing a form, or calling a script function defined by the document author. for example.
 - **Management of form GUI:** If a web page contains a form with fill-in fields, the browser must allow the user to perform standard text-editing functions within these fields. It also needs to automatically provide certain graphical feedback, such as changing a button image when it is pressed or providing a text cursor in a text field that will receive keyboard input.
 - **Secure communication:** When the user sends sensitive information, such as a credit card number, to a web server, the browser can encode this information in a way the prevents any machines along the IP route from the client to the server from obtaining the information.
 - **Plug-in execution:** While the browser itself normally understands only a limited number of MIME types, most browsers support some form of plug-in protocol that allows the browser’s capabilities to be supplemented by other software. If a browser has a plugin for displaying, say, a document conforming to the application/pdf MIME type, then when the browser receives such a document it will pass it—via the plug-in protocol—to the appropriate plug-in for display. Some plug-ins may display the document within the browser’s client area, while others may display in a separate window that is controlled by the plug-in itself. Plug-ins are often installed automatically, after user permission is obtained, when an unsupported MIME type is encountered. To see a list of plug-ins installed in your copy of Mozilla, select **Help|About Plug-ins**.

1.7 WEB SERVERS

1.7.1 Server Features

- The primary feature of every web server is to accept HTTP requests from web clients and return an appropriate resource (if available) in the HTTP response. Even this basic functionality involves a number of steps .
- The server calls on TCP software and waits for connection requests to one or more ports.
- When a connection request is received, the server dedicates a “subtask” to handling this connection.
- The subtask establishes the TCP connection and receives an HTTP request.
- The subtask examines the Host header field of the request to determine which “virtual host” should receive this request and invokes software for this host.
- The virtual host software maps the Request-URI field of the HTTP request start line to a resource on the server.
- If the resource is a file, the host software determines the MIME type of the file and creates an HTTP response that contains the file in the body of the response message.
- If the resource is a program, the host software runs the program, providing it with information from the request and returning the output from the program as the body of an HTTP response message.
- The server normally logs information about the request and response—such as the IP address of the requester and the status code of the response—in a plain-text file
- If the TCP connection is kept alive, the server subtask continues to monitor the connection until a certain length of time has elapsed, the client sends another request, or the client initiates a connection close.

1.7.2 Server History

- Both servers can be configured to run a variety of types of programs, although certain programming languages tend to be used more frequently on one system than the other.
- For example, many IIS servers run programs written in VBScript (a derivative of Visual Basic), while a typical Apache server might run programs written in either Perl or the PHP scripting language (PHP stands for “PHP Hypertext Processor”; yes, the definition is infinitely recursive).
- A number of IIS and Apache servers also run Java programs. When running a Java program, both Apache and IIS servers are usually configured to run the program by using separate software called a servlet container. The servlet container provides the Java Virtual Machine that runs the Java program (known as a servlet), and also provides communication between the servlet and the Apache or IIS web server.

1.7.3 Server Configuration and Tuning

- Broadly speaking, server configuration can be broken into two areas: external communication and internal processing. In Tomcat, this corresponds to two separate Java packages: Coyote, which provides the HTTP/1.1 communication, and Catalina, which is the actual servlet container. Some of the Coyote parameters, affecting external communication, include the following:
 - IP addresses and TCP ports that may be used to connect to this server.

- Number of subtasks (called threads in Java) that will be created when the server is initialized. This many TCP connections can be established simultaneously with minimal overhead.
 - Maximum number of threads that will be allowed to exist simultaneously. If this is larger than the previous value, then the number of threads maintained by the server may change, either up or down, over time.
 - Maximum number of TCP connection requests that will be queued if the server is already running its maximum number of threads. Connection requests received if the queue is full will be refused.
 - Length of time the server will wait after serving an HTTP request over a TCP connection before closing the connection if another request is not received.
 - The settings of these parameters can have a significant influence on the performance of a server; changing the values of these and similar parameters in order to optimize performance is often referred to as tuning the server.
- Load generation or stress test tools can be used to simulate requests to a web server, and can therefore be helpful for experimenting with tuning parameters based on anticipated traffic patterns even before a web site “goes live.”
- The internal Catalina portion of Tomcat also has a number of parameter settings that affect functionality. These settings can determine:
 - Which client machines may send HTTP requests to the server.
 - Which virtual hosts are listening for TCP connections on a given port.
 - What logging will be performed.
 - How the path portion of Request-URIs will be mapped to the server’s file system or other resources.
 - Whether or not the server’s resources will be password protected.
 - Whether or not resources will be cached in the server’s memory.

Field Name	Description
Accept Count	Length of the TCP connection wait queue.
Connection Timeout	Server will close connection if it is idle for this many milliseconds.
IP Address	Blank indicates that this Connector will accept TCP connections directed to any IP address associated with this machine. Specifying an address restricts connections to requests for that address.
Port Number	Port number on which this Connection will listen for TCP connection requests.
Min Spare Threads	Initial number of threads that will be allocated to process TCP connections associated with this Connector. Once connections are established with the Connector, the server will maintain at least this many <i>idle</i> processing threads, that is, threads waiting for new connections but otherwise unused.
Max Threads	Maximum number of threads that will be allocated to process TCP connections associated with this Connector.
Max Spare Threads	Maximum number of idle threads allowed to exist at any one time. The server will begin stopping threads if the number of idle threads exceeds this value.

Table 1.10 Some Of The Fields For The Connector Component

1.7.4 Defining Virtual Hosts

- The virtual host name should normally be a fully qualified domain name that would be used by visitors to your web site, although the Host supplied as part of the JWSDP Service is given the unqualified name localhost.

- This is a special name that the DNS system treats as a reference to a special IP address, 127.0.0.1. If an IP message is sent to this address, the IP software causes the message to loop back to itself for receipt.
- In short, browsing to a URL with domain name localhost causes the browser to send the HTTP request to a web server on the machine running the browser.

Field Name	Description
Name	Usually the fully qualified domain name (or <code>localhost</code>) that clients will use to access this virtual host.
Application Base	Directory containing <i>web applications</i> for this virtual host (see text).
Deploy on Startup	Boolean value indicating whether or not web applications should be automatically initialized when the server starts.
Auto Deploy	Boolean value indicating whether or not web applications added to the Application Base while the server is running should be automatically initialized.

Table 1.11 Key Fields for Host Component

1.7.5 Logging

- Web server logs record information about server activity. The primary web server log recording normal activity is an access log, a file that records information about every HTTP request processed by the server.
- A web server may also produce one or more message logs containing a variety of debugging and other information generated by web applications as well as possibly by the web server itself.
- Finally, information written to the standard output and error streams by the web server or applications may also be logged

Field Name	Description
Directory	Directory (relative to Tomcat installation directory or absolute) where log file will be written
Pattern	Information to be written to the log (see text)
Prefix	String that will be used to begin log file name
Resolve Hosts	Whether IP addresses (<code>False</code> value) or host names (<code>True</code> value) should be written to the log file
Rotatable	Whether or not date should be added to file name and file should be automatically rotated each day
Suffix	String that will be used to end log file name

Table 1.12 Key Fields for Valve Component of Type AccessLogValve

- The combination of values for the Directory, Prefix, Rotatable, and Suffix fields determine the file system path to the access log. The JWSDP Service settings for the values of these Valve fields cause the access log for this Service to be written to the logs directory under the JWSDP 1.3 installation directory in a file that starts with the string `access.log.` and ends with the string `.txt`. In between these strings, because Rotatable is given the value `True`, Tomcat inserts the current date, in `YYYY-MM-DD` (year-month-day) format.
- The Pattern for the JWSDP Service access log Valve is `%h %l %u %t "%r" %s %b` This corresponds to what is often called the common access log format
- The following information is contained in this log entry:
 - Host name (or IP address; see Table 1.12) of client machine making the request.

- User name used to log in, if server password protection is enabled.
- Date and time of response, plus the time zone (offset from GMT) of the time.
- Start line of HTTP request (quoted)
- HTTP status code of response
- Number of bytes sent in body of response

1.7.6 Access Control

- Tomcat can provide automatic password protection for resources that it serves. At its heart, this is a two-stage process.
- First, a database of user names is created. Each user name is assigned a password and a list of roles.
- The second stage of this process—associating resources with required roles—is normally performed by web application developers.
- The first stage—defining one or more user databases—can be performed by web system administrators, application developers, or both. The JWSDP Service contains an example of a database defined at the Service level through the use of a Realm component, which associates a user database with a Service.
- A coarser-grained access control can be provided by using Valve objects of type RemoteHostValve and RemoteAddressValve. Both are used to specify client machines that should be rejected if they request a connection to the server. They differ only in whether client machine host names or IP addresses are specified. Each type of Valve has two possible lists of clients: an Allow list and a Deny list. If one or more host names (comma-separated) is entered in the Allow list, then only these hosts can access the server.

1.7.7 Secure Servers

- In general, any machine other than the sender or receiver that extracts information from network messages is known as an eavesdropper.
- To prevent eavesdroppers from obtaining sensitive information, such as credit card numbers, all such sensitive information should be encrypted before being transmitted over any public communication network. The standard means of indicating to a browser that it should encrypt an HTTP request is to use the https scheme on the URL for the request. For example, entering the URL <https://www.example.org> in Mozilla's Location bar will cause the browser to attempt to send an encrypted HTTP GET request to www.example.org.
- A client browser that wishes to communicate securely with a server begins by initiating (over TCP/IP) a TLS Handshake with the server. During the Handshake process, the server and client agree on various parameters that will be used to encrypt messages sent between them. The server also sends a certificate to the client. The certificate enables the client to be sure that the machine it is communicating with is the one the client intends (as specified by the host name in the URL the browser is requesting). Certificates are necessary to avoid so-called man-in-the-middle attacks, in which some machine intercepts a message intended for another machine (the target), prevents the message from further forwarding, and returns an HTTP reply to the sender pretending to be from the target.
- Tomcat supports the TLS 1.0 and earlier protocols. To enable the secure server Tomcat features, you must do two things:
 - Obtain and install a certificate.
 - Configure the server to listen for TLS connections on some port.

1.8 HTML5

- HTML stands for Hyper Text Markup Language. It is used to design web pages using markup language.
- HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages.
- Markup language is used to define the text document within tag which defines the structure of web pages. HTML 5 is the fifth and current version of HTML.
- It has improved the markup available for documents and has introduced application programming interfaces(API) and Document Object Model(DOM).

Features:

- It has introduced new multimedia features which supports audio and video controls by using `<audio>` and `<video>` tags.
- There are new graphics elements including vector graphics and tags.
- Enrich semantic content by including `<header>` `<footer>`, `<article>`, `<section>` and `<figure>` are added.
- Drag and Drop- The user can grab an object and drag it further dropping it on a new location.
- Geo-location services- It helps to locate the geographical location of a client.
- Web storage facility which provides web application methods to store data on web browser.
- Uses SQL database to store data offline.
- Allows to draw various shapes like triangle, rectangle, circle, etc.
- Capable of handling incorrect syntax.
- Easy DOCTYPE declaration i.e. `<!doctype html>`
- Easy character encoding i.e. `<meta charset="UTF-8">`

Removed elements from HTML 5: There are many elements which are depreciated from HTML 5 are listed below:

Removed Elements	Use Instead Elements
<code><acronym></code>	<code><abbr></code>
<code><applet></code>	<code><object></code>
<code><basefont></code>	CSS
<code><big></code>	CSS
<code><center></code>	CSS
<code><dir></code>	<code></code>
<code></code>	CSS
<code><frame></code>	
<code><frameset></code>	
<code><noframes></code>	
<code><isindex></code>	
<code><strike></code>	CSS, <code><s></code> or <code></code>
<code><tt></code>	CSS

New Added Elements in HTML 5:

- **<article>:** The `<article>` tag is used to represent an article. More specifically, the content within the `<article>` tag is independent from the other content of the site (even though it can be related).

- **<aside>**: The <aside> tag is used to describe the main object of the web page in a shorter way like a highlighter. It basically identifies the content that is related to the primary content of the web page but does not constitute the main intent of the primary page. The <aside> tag contains mainly author information, links, related content and so on.
- **<figcaption>**: The <figurecaption> tag in HTML is used to set a caption to the figure element in a document.
- **<figure>**: The <figure> tag in HTML is used to add self-contained content like illustrations, diagrams, photos or codes listing in a document. It is related to main flow but it can be used in any position of a document and the figure goes with the flow of the document and if remove it then it should not affect the flow of the document.
- **<header>**: It contains the section heading as well as other content, such as a navigation links, table of contents, etc.
- **<footer>**: The <footer> tag in HTML is used to define a footer of HTML document. This section contains the footer information (author information, copyright information, carriers etc). The footer tag are used within body tag. The <footer> tag is new in the HTML 5. The footer elements require a start tag as well as an end tag.
- **<main>**: Delineates the main content of the body of a document or web app.
- **<mark>**: The <mark> tag in HTML is used to define the marked text. It is used to highlight the part of the text in the paragraph.
- **<nav>**: The <nav> tag is used to declaring the navigational section in HTML documents. Websites typically have sections dedicated to navigational links, which enables user to navigate the site. These links can be placed inside a nav tag.
- **<section>**: It demarcates a thematic grouping of content.
- **<details>**: The <details> tag is used for the content/information which is initially hidden but could be displayed if the user wishes to see it. This tag is used to create interactive widget which user can open or close it. The content of details tag is visible when open the set attributes.
- **<summary>**: The <summary> tag in HTML is used to define a summary for the <details> element. The <summary> element is used along with the <details> element and provides a summary visible to the user. When the summary is clicked by the user, the content placed inside the <details> element becomes visible which was previously hidden. The <summary> tag was added in HTMI 5. The <summary> tag requires both starting and ending tag.
- **<time>**: The <time> tag is used to display the human-readable data/time. It can also be used to encode dates and times in a machine-readable form. The main advantage for users is that they can offer to add birthday reminders or scheduled events in their calender's and search engines can produce smarter search results.
- **<bdi>**: The <bdi> tag refers to the Bi-Directional Isolation. It differentiate a text from other text that may be formatted in different direction. This tag is used when a user generated text with an unknown directions.
- **<wbr>**: The <wbr> tag in HTML stands for word break opportunity and is used to define the position within the text which is treated as a line break by the browser. It is mostly used when the used word is too long and there are chances that the browser may break lines at the wrong place for fitting the text.
- **<datalist>**: The <datalist> tag is used to provide autocomplete feature in the HTML files. It can be used with input tag, so that users can easily fill the data in the forms using select the data.

- **<keygen>**: The <keygen> tag in HTML is used to specify a key-pair generator field in a form. The purpose of <keygen> element is to provide a secure way to authenticate users. When a form is submitted then two keys are generated, private key and public key. The private key stored locally, and the public key is sent to the server. The public key is used to generate client certificate to authenticate user for future.
- **<output>**: The <output> tag in HTML is used to represent the result of a calculation performed by the client-side script such as JavaScript.
- **<progress>**: It is used to represent the progress of a task. It is also define that how much work is done and how much is left to download a things. It is not used to represent the disk space or relevant query.
- **<svg>**: It is the Scalable Vector Graphics.
- **<canvas>**: The <canvas> tag in HTML is used to draw graphics on web page using JavaScript. It can be used to draw paths, boxes, texts, gradient and adding images. By default it does not contains border and text.
- **<audio>**: It defines the music or audio content.
- **<embed>**: Defines containers for external applications (usually a video player).
- **<source>**: It defines the sources for <video> and <audio>.
- **<track>**: It defines the tracks for <video> and <audio>.
- **<video>**: It defines the video content.

Advantages:

- All browsers supported.
- More device friendly.
- Easy to use and implement.
- HTML 5 in integration with CSS, JavaScript, etc can help build beautiful websites.

Disadvantages:

- Long codes have to be written which is time consuming.
- Only modern browsers support it.

1.9 TABLES

- HTML provides a fairly sophisticated model for presenting data in tabular form.
- Columns and rows will automatically size to contain their data, although there are also various ways to specify column widths; individual table cells can span multiple rows and/or columns; header and/or footer rows can be supplied; and so on. There are also various options for changing the visual appearance of a table, such as the widths of its internal cell-separating lines (rules) and external borders.
- Simple tables are simple to represent in HTML. For example, a table of student grades could be written as follows and produces the table shown in Figure 1.1

```

<table border="5">
<tr>
<td>Kim</td><td>100</td><td>89</td>
</tr>
<tr>
<td>Sandy</td><td>78</td><td>92</td>
</tr>
<tr>
<td>Taylor</td><td>83</td><td>73</td>
</tr>
</table>

```

- A tr (table row) element is used to contain each row. Within a row, a td (table data) element marks each element of the row. Notice that we don't need to specify the number of rows and columns in the table explicitly. Instead, these values are determined automatically: in a simple table, the number of rows is determined by the number of tr elements in the table, and the number of columns is determined by the maximum number of td elements contained within any row

The screenshot shows a simple table with 3 rows and 3 columns. The first column contains student names: 'Kim', 'Sandy', and 'Taylor'. The second column contains scores: '100', '78', and '83'. The third column contains scores: '89', '92', and '73'. The table has a border of 1 pixel.

Kim	100	89
Sandy	78	92
Taylor	83	73

Fig 1.16 A simple table of grades.

- In this example, since there are three tr elements, each containing three td elements, the table is 3 by 3. Finally, notice that the width of table columns is also automatically adjusted to contain the maximum width item in any column, although this can be overridden via the style attribute

```
<table border="5">
<caption>
COSC 400 Student Grades
</caption>
<tr>
<td>&nbsp;</td><td>&nbsp;</td><th colspan="2">Grades</th>
</tr>
<tr>
<td>&nbsp;</td><th>Student</th><th>Exam 1</th><th>Exam 2</th>
</tr>
<tr>
<th rowspan="2">Undergraduates</th><td>Kim</td><td>100</td><td>89</td>
</tr>
<tr>
<td>Sandy</td><td>78</td><td>92</td>
</tr>
<tr>
<th>Graduates</th><td>Taylor</td><td>83</td><td>73</td>
</tr>
</table>
```

The screenshot shows a table with a header row and two data rows. The header row contains four columns: 'Student', 'Exam 1', and 'Exam 2'. The first data row contains 'Undergraduates' in the first column and student information in the other three columns. The second data row contains 'Graduates' in the first column and student information in the other three columns. The table has a border of 1 pixel.

		Grades	
	Student	Exam 1	Exam 2
Undergraduates	Kim	100	89
	Sandy	78	92
Graduates	Taylor	83	73

Fig.1.2 Table with headings and caption.

- Two new elements are used in this example. The caption element, as the implies, is used to define a caption for table. If a caption element is used with a the caption start tag must appear immediately after the start tag of the element.
- The second new element, th (table header), is much like the td element, except that a typical browser will format the content of a th element in boldface and center it horizontally within the column.

1.10 LISTS

- Fig 1.3 illustrates the three types of lists supported by HTML:
 - Unordered: A bullet list
 - Ordered: A numbered list
 - Definition: A list of terms and definitions for each
 - This figure was produced by the following HTML:

```
<ul>
<li>Bulleted list item</li>
<li>Bulleted list item 2</li>
</ul>

<ol>
<li>Numbered list item</li>
<li>Numbered list item 2</li>
</ol>

<dl>
<dt>Term</dt>
<dd>Definition of term</dd>
<dt>Term 2</dt>
<dd>Definition of term 2</dd></dl>
```

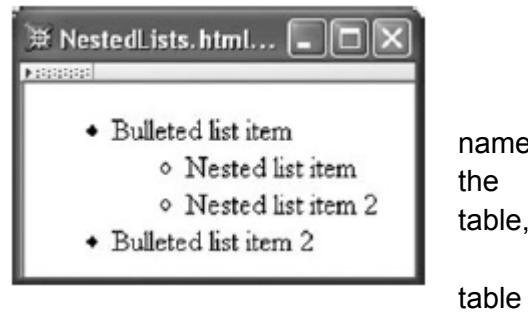
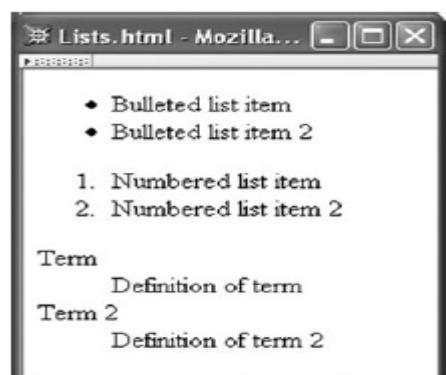


Fig1.4 Browser rendering nested unordered lists.

- Lists can be nested to produce an outline layout. For example, the markup

```
<ul>
<li>Bulleted list item
<ul>
<li>Nested list item</li>
<li>Nested list item 2</li>
</ul>
</li>
<li>Bulleted list item 2</li>
</ul>
```



1.11 IMAGES

Images can improve the design and the appearance of a web page.

HTML Images Syntax

- In HTML, images are defined with the `` tag.
- The `` tag is empty, it contains attributes only, and does not have a closing tag.
- The `src` attribute specifies the URL (web address) of the image:

``

The alt Attribute

- The `alt` attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the `src` attribute, or if the user uses a screen reader).
- The value of the `alt` attribute should describe the image:

Example

``

Image Size - Width and Height

The `style` attribute to specify the width and height of an image.

Example

``

Note: Always specify the width and height of an image. If width and height are not specified, the page might flicker while the image loads. Width and Height, or Style? **The width, height, and style attributes are valid in HTML.** However, we suggest using the `style` attribute. It prevents styles sheets from changing the size of images:

Example

```
<!DOCTYPE.html>
<html>
<head>
<style>
img{
  width:100%;
}
</style>
</head>
<body>


</body>
</html>
```

Images in Another Folder

If not specified, the browser expects to find the image in the same folder as the web page.

However, it is common to store images in a sub-folder. You must then include the folder name in the `src` attribute:

Example

``

Images on Another Server

- Some web sites store their images on image servers.
- Actually, you can access images from any web address in the world:

Example

``

Animated Images

HTML allows animated GIFs:

Example

```

```

Image as a Link

To use an image as a link, put the `` tag inside the `<a>` tag:

Example

```
<a href="default.asp">  
      
</a>
```

Image Floating

Use the CSS float property to let the image float to the right or to the left of a text:

Example

```
<p>
```

The image will float to the right of the text.</p>

```
<p>
```

The image will float to the left of the text.</p>

HTML Image Tags

Tag	Description
-----	-------------

<code></code>	Defines an image
--------------------------	------------------

<code><map></code>	Defines an image-map
--------------------------	----------------------

<code><area></code>	Defines a clickable area inside an image-map
---------------------------	--

<code><picture></code>	Defines a container for multiple image resources
------------------------------	--

Image Maps

- The `<map>` tag defines an image-map. An image-map is an image with clickable areas.
- Click on the computer, the phone, or the cup of coffee in the image below:

Example

```
  
<mapname="workmap">  
    <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">  
    <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">  
    <area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">  
</map>
```

The Image

The image is inserted using the `` tag. The only difference from other images is that you must add a `usemap` attribute:

```

```

The `usemap` value starts with a hash tag `#` followed by the name of the image map, and is used to create a relationship between the image and the image map.

1.12 HTML5 CONTROL ELEMENTS

- An HTML element usually consists of a **start** tag and an **end** tag, with the content inserted in between:

```
<tagname>Content goes here...</tagname>
```

- The HTML **element** is everything from the start tag to the end tag:

```
<p>My first paragraph.</p>
```

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
<p>	My first paragraph.	</p>

1.12.1 Nested HTML Elements

- HTML elements can be nested (elements can contain elements).
- All HTML documents consist of nested HTML elements.
- This example contains four HTML elements:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>


- The <html> element defines the whole document.
- It has a start tag <html> and an end tag </html>.
- Inside the <html> element is the <body> element.

```

1.12.2 HTML New Tag Elements

Tags (Elements)	Description
<article>	Represents an independent piece of content of a document, such as a blog entry or newspaper article
<aside>	Represents a piece of content that is only slightly related to the rest of the page.
<audio>	Defines an audio file.
<canvas>	This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<command>	Represents a command the user can invoke.
<datalist>	Together with the a new list attribute for input can be used to make comboboxes
<details>	Represents additional information or controls which the user can obtain on demand

<embed>	Defines external interactive content or plugin.
<figure>	Represents a piece of self-contained flow content, typically referenced as a single unit from the main flow of the document.
<footer>	Represents a footer for a section and can contain information about the author, copyright information, et cetera.
<header>	Represents a group of introductory or navigational aids.
<hgroup>	Represents the header of a section.
<keygen>	Represents control for key pair generation.
<mark>	Represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.
<meter>	Represents a measurement, such as disk usage.
<nav>	Represents a section of the document intended for navigation.
<output>	Represents some type of output, such as from a calculation done through scripting.
<progress>	Represents a completion of a task, such as downloading or when performing a series of expensive operations.
<ruby>	Together with <rt> and <rp> allow for marking up ruby annotations.
<section>	Represents a generic document or application section
<time>	Represents a date and/or time.
<video>	Defines a video file.
<wbr>	Represents a line break opportunity.

- New types for <input> tag
- The input element's type attribute now has the following new values –

Type	Description
color	Color selector, which could be represented by a wheel or swatch picker
date	Selector for calendar date
datetime-local	Date and time display, with no setting or indication for time zones
datetime	Full date and time display, including a time zone.
email	Input type should be an email.
month	Selector for a month within a given year
number	A field containing a numeric value only
range	Numeric selector within a range of values, typically visualized as a slider
search	Term to supply to a search engine. For example, the search bar atop a browser.
tel	Input type should be telephone number.
time	Time indicator and selector, with no time zone information
url	Input type should be URL type.
week	Selector for a week within a given year

1.13 SEMANTIC ELEMENTS

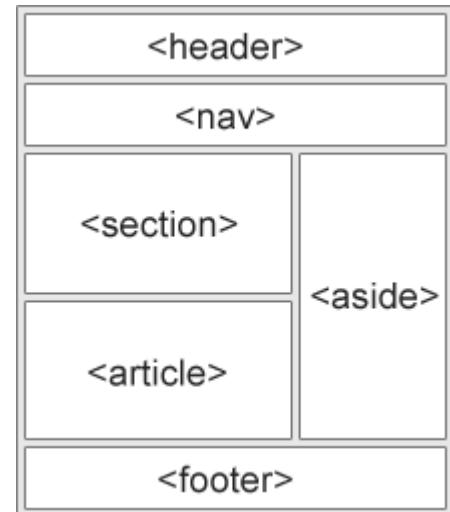
- Semantics is the study of the meanings of words and phrases in a language.
- Semantic elements = elements with a meaning.

What are Semantic Elements?

- A semantic element clearly describes its meaning to both the browser and the developer.
- Examples of non-semantic elements: `<div>` and `` - Tells nothing about its content.
- Examples of semantic elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

New Semantic Elements in HTML5

- Many web sites contain HTML code like: `<div id="nav">` `<div class="header">` `<div id="footer">` to indicate navigation, header, and footer.
- HTML5 offers new semantic elements to define different parts of a web page:
 - `<article>`
 - `<aside>`
 - `<details>`
 - `<figcaption>`
 - `<figure>`
 - `<footer>`
 - `<header>`
 - `<main>`
 - `<mark>`
 - `<nav>`
 - `<section>`
 - `<summary>`
 - `<time>`



HTML5 `<section>` Element

- The `<section>` element defines a section in a document.
- According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."
- A home page could normally be split into sections for introduction, content, and contact information.

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>
```

HTML5 `<article>` Element

- The `<article>` element specifies independent, self-contained content.
- An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.
- Examples of where an `<article>` element can be used:
 - Forum post
 - Blog post
 - Newspaper article

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural
```

```
environment,  
    and build a future in which humans live in harmony with nature.</p>  
</article>
```

Nesting <article> in <section> or Vice Versa?

- The <article> element specifies independent, self-contained content.
- The <section> element defines section in a document.

HTML5 <header> Element

- The <header> element specifies a header for a document or section.
- The <header> element should be used as a container for introductory content.
- The following example defines a header for an article:

Example

```
<article>  
  <header>  
    <h1>What Does WWF Do?</h1>  
    <p>WWF's mission:</p>  
  </header>  
  <p>WWF's mission is to stop the degradation of our planet's natural environment,  
  and build a future in which humans live in harmony with nature.</p>  
</article>
```

HTML5 <footer> Element

- The <footer> element specifies a footer for a document or section.
- A <footer> element should contain information about its containing element.
- A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

Example

```
<footer>  
  <p>Posted by: Hege Refsnes</p>  
  <p>Contact information: <a href="mailto:someone@example.com">  
  someone@example.com</a>.</p>  
</footer>
```

HTML5 <nav> Element

- The <nav> element defines a set of navigation links.

Example

```
<nav>  
  <a href="/html/">HTML</a> |  
  <a href="/css/">CSS</a> |  
  <a href="/js/">JavaScript</a> |  
  <a href="/jquery/">jQuery</a>  
</nav>
```

HTML5 <aside> Element

- The <aside> element defines some content aside from the content it is placed in (like a sidebar).
- The <aside> content should be related to the surrounding content.

Example

```
<p>My family and I visited The Epcot center this summer.</p>
```

```
<aside>  
  <h4>Epcot Center</h4>
```

```
<p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

HTML5 <figure> and <figcaption> Elements

- The purpose of a figure caption is to add a visual explanation to an image.
- In HTML5, an image and a caption can be grouped together in a <figure> element:

Example

```
<figure>
  
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

Why Semantic Elements?

- With HTML4, developers used their own id/class names to style elements: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, etc.
- This made it impossible for search engines to identify the correct web page content.
- With the new HTML5 elements (<header> <footer> <nav> <section> <article>), this will become easier.
- According to the W3C, a Semantic Web: "Allows data to be shared and reused across applications, enterprises, and communities."

Semantic Elements in HTML5

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

1.14 HTML5 DRAG AND DROP

- Drag and Drop (DnD) is powerful User Interface concept which makes it easy to copy, reorder and deletion of items with the help of mouse clicks. This allows the user to click and hold the mouse button down over an element, drag it to another location, and release the mouse button to drop the element there.
- To achieve drag and drop functionality with traditional HTML4, developers would either have to either have to use complex JavaScript programming or other JavaScript frameworks like jQuery etc.
- Now HTML 5 came up with a Drag and Drop (DnD) API that brings native DnD support to the browser making it much easier to code up.

- HTML 5 DnD is supported by all the major browsers like Chrome, Firefox 3.5 and Safari 4 etc.

Drag and Drop Events

There are number of events which are fired during various stages of the drag and drop operation. These events are listed below –

S.No	Events & Description
1	Dragstart Fires when the user starts dragging of the object.
2	Dragenter Fired when the mouse is first moved over the target element while a drag is occurring. A listener for this event should indicate whether a drop is allowed over this location. If there are no listeners, or the listeners perform no operations, then a drop is not allowed by default.
3	Dragover This event is fired as the mouse is moved over an element when a drag is occurring. Much of the time, the operation that occurs during a listener will be the same as the dragenter event.
4	Dragleave This event is fired when the mouse leaves an element while a drag is occurring. Listeners should remove any highlighting or insertion markers used for drop feedback.
5	Drag Fires every time the mouse is moved while the object is being dragged.
6	Drop The drop event is fired on the element where the drop was occurred at the end of the drag operation. A listener would be responsible for retrieving the data being dragged and inserting it at the drop location.
7	Dragend Fires when the user releases the mouse button while dragging an object.

The DataTransfer Object

- The event listener methods for all the drag and drop events accept Event object which has a readonly attribute called dataTransfer.
- The event.dataTransfer returns DataTransfer object associated with the event as follows –

```
function EnterHandler(event) {
    DataTransfer dt = event.dataTransfer;
    .....
}
```

- The *DataTransfer* object holds data about the drag and drop operation. This data can be retrieved and set in terms of various attributes associated with DataTransfer object as explained below –

Sr.No	DataTransfer attributes and their description
1	<code>dataTransfer.dropEffect [= value]</code> Returns the kind of operation that is currently selected. This attribute can be set, to change the selected operation. The possible values are none, copy, link, and move.
2	<code>dataTransfer.effectAllowed [= value]</code> Returns the kinds of operations that are to be allowed. This attribute can be set, to change the allowed operations. The possible values are none, copy, copyLink, copyMove, link, linkMove, move, all and uninitialized.
3	<code>dataTransfer.types</code> Returns a DOMStringList listing the formats that were set in the dragstart event. In addition, if any files are being dragged, then one of the types will be the string "Files".
4	<code>dataTransfer.clearData ([format])</code> Removes the data of the specified formats. Removes all data if the argument is omitted.
5	<code>dataTransfer.setData(format, data)</code> Adds the specified data.
6	<code>data = dataTransfer.getData(format)</code> Returns the specified data. If there is no such data, returns the empty string.
7	<code>dataTransfer.files</code> Returns a FileList of the files being dragged, if any.

Drag and Drop Process

- Following are the steps to be carried out to implement Drag and Drop operation –

Step 1 - Making an Object Draggable

Here are steps to be taken –

- If you want to drag an element, you need to set the **draggable** attribute to **true** for that element.
- Set an event listener for **dragstart** that stores the data being dragged.
- The event listener **dragstart** will set the allowed effects (copy, move, link, or some combination).

```
<!DOCTYPE HTML>
<html>
  <head>
    <style type = "text/css">
      #boxA, #boxB {
```

```

float:left;padding:10px;margin:10px; -moz-user-select:none;
}

#boxA { background-color: #6633FF; width:75px; height:75px; }

#boxB { background-color: #FF6699; width:150px; height:150px; }

</style>

<script type = "text/javascript">

    function dragStart(ev) {

        ev.dataTransfer.effectAllowed = 'move';

        ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));

        ev.dataTransfer.setDragImage(ev.target,0,0);

        return true;

    }

</script>

</head>

<body>

    <center>

        <h2>Drag and drop HTML5 demo</h2>

        <div>Try to drag the purple box around.</div>

        <div id = "boxA" draggable = "true"

            ondragstart = "return dragStart(ev)">

            <p>Drag Me</p>

        </div>

        <div id = "boxB">Dustbin</div>

    </center>

</body>

</html>

```

Step 2 - Dropping the Object

To accept a drop, the drop target has to listen to at least three events.

- The **dragenter** event, which is used to determine whether or not the drop target is to accept the drop. If the drop is to be accepted, then this event has to be canceled.
- The **dragover** event, which is used to determine what feedback is to be shown to the user. If the event is canceled, then the feedback (typically the cursor) is updated based on the dropEffect attribute's value.

- Finally, the **drop** event, which allows the actual drop to be performed.

Live Demo

```
<html>
<head>
<style type="text/css">
#boxA, #boxB {
    float:left;padding:10px;margin:10px;-moz-user-select:none;
}
#boxA { background-color: #6633FF; width:75px; height:75px; }
#boxB { background-color: #FF6699; width:150px; height:150px; }
</style>
<script type="text/javascript">
function dragStart(ev) {
    ev.dataTransfer.effectAllowed='move';
    ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
    ev.dataTransfer.setDragImage(ev.target,0,0);
    return true;
}
function dragEnter(ev) {
    event.preventDefault();
    return true;
}
function dragOver(ev) {
    return false;
}
function dragDrop(ev) {
    var src = ev.dataTransfer.getData("Text");
    ev.target.appendChild(document.getElementById(src));
    ev.stopPropagation();
    return false;
}
</script>
```

```

</script>
</head>
<body>
<center>
<h2>Drag and drop HTML5 demo</h2>
<div>Try to move the purple box into the pink box.</div>
<div id="boxA" draggable="true" ondragstart="return dragStart(event)">
<p>Drag Me</p>
</div>
<div id="boxB" ondragenter="return dragEnter(event)" ondrop="return dragDrop(event)" ondragover="return dragOver(event)">Dustbin</div>
</center> </body>
</html>

```

1.15 HTML AUDIO/VIDEO DOM

HTML Audio and Video DOM Reference

- The HTML5 DOM has methods, properties, and events for the `<audio>` and `<video>` elements.
- These methods, properties, and events allow you to manipulate `<audio>` and `<video>` elements using JavaScript.

HTML Audio/Video Methods

Method	Description
<code>addTextTrack()</code>	Adds a new text track to the audio/video
<code>canPlayType()</code>	Checks if the browser can play the specified audio/video type
<code>load()</code>	Re-loads the audio/video element
<code>play()</code>	Starts playing the audio/video
<code>pause()</code>	Pauses the currently playing audio/video

HTML Audio/Video Properties

Property	Description

audioTracks	Returns an AudioTrackList object representing available audio tracks Autoplay
buffered	Returns a TimeRanges object representing the buffered parts of the audio/video
controller	Returns the MediaController object representing the current media controller of the audio/video
controls	Sets or returns whether the audio/video should display controls (like play/pause etc.)
currentTime	Returns the URL of the current audio/video
defaultMuted	Sets or returns whether the audio/video should be muted by default

HTML Audio/Video Events

Event	Description
error	Fires when an error occurred during the loading of an audio/video
loadeddata	Fires when the browser has loaded the current frame of the audio/video
loadedmetadata	Fires when the browser has loaded meta data for the audio/video
loadstart	Fires when the browser starts looking for the audio/video
pause	Fires when the audio/video has been paused
play	Fires when the audio/video has been started or is no longer paused
playing	Fires when the audio/video is playing after having been paused or stopped for buffering
progress	Fires when the browser is downloading the audio/video
ratechange	Fires when the playing speed of the audio/video is changed
seeked	Fires when the user is finished moving/skipping to a new position in the audio/video
seeking	Fires when the user starts moving/skipping to a new position in the audio/video
stalled	Fires when the browser is trying to get media data, but data is not available
suspend	Fires when the browser is intentionally not getting media data
timeupdate	Fires when the current playback position has changed

volumechange	Fires when the volume has been changed
waiting	Fires when the video stops because it needs to buffer the next frame

1.16 Css3

Types of CSS (Cascading Style Sheet)

- Cascading Style Sheet(CSS) is used to set the style in web pages which contain HTML elements. It sets the background color, font-size, font-family, color, ... etc property of elements in a web pages.
- There are three types of CSS which are given below:
 - Inline CSS
 - Internal or Embedded CSS
 - External CSS

Inline CSS:

Inline CSS contains the CSS property in the body section attached with element is known as inline CSS. This kind of style is specified within an HTML tag using style attribute.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline CSS</title>
  </head>

  <body>
    <p style = "color:#009900; font-size:50px;
      font-style:italic; text-align:center;">
      GeeksForGeeks
    </p>
  </body>
</html>
```

Internal or Embedded CSS:

This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Internal CSS</title>
    <style>
      .main {
        text-align:center;
      }
      .GFG {
        color:#009900;
        font-size:50px;
        font-weight:bold;
      }
      .geeks {
        font-style:bold;
        font-size:20px;
      }
    </style>
  </head>
  <body>
    <h1 class="main">GeeksforGeeks</h1>
    <div class="GFG">GeeksforGeeks</div>
    <h2 class="geeks">GeeksforGeeks</h2>
  </body>
</html>
```

```

        }
    </style>
</head>
<body>
    <div class = "main">
        <div class ="GFG">GeeksForGeeks</div>

        <div class ="geeks">
            A computer science portal for geeks
        </div>
    </div>
</body>
</html>

```

External CSS:

External CSS contains separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, ... etc). CSS property written in a separate file with .css extension and should be linked to the HTML document using **link** tag. This means that for each element, style can be set only once and that will be applied across web pages.

Example: The file given below contains CSS property. This file save with .css extension. For Ex: **geeks.css**

```

body {
    background-color:powderblue;
}
.main {
    text-align:center;
}
.GFG {
    color:#009900;
    font-size:50px;
    font-weight:bold;
}
#geeks {
    font-style:bold;
    font-size:20px;
}

```

Below is the HTML file that is making use of the created external style sheet

- **link** tag is used to link the external style sheet with the html webpage.
- **href** attribute is used to specify the location of the external style sheet file.

filter_none

edit

play_arrow

```

brightness_4
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="geeks.css"/>
    </head>

```

```

<body>
  <div class = "main">
    <div class ="GFG">GeeksForGeeks</div>
    <div id ="geeks">
      A computer science portal for geeks
    </div>
  </div>
</body>
</html>

```

Properties of CSS: Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority. Multiple style sheets can be defined on one page. If for an HTML tag, styles are defined in multiple style sheets then the below order will be followed.

- As Inline has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles.
- Internal or Embedded stands second in the priority list and overrides the styles in the external style sheet.
- External style sheets have the least priority. If there are no styles defined either in inline or internal style sheet then external style sheet rules are applied for the HTML tags.

Inheritance and cascade

Inheritance and the cascade are two fundamental concepts in CSS, that are important to understand. The two concepts are closely related, yet different:

- Inheritance is associated with how the elements in the HTML markup inherit properties from their parent (containing) elements and pass them on to their children.
- The cascade relates to CSS declarations being applied to a document, and how conflicting rules do or do not override each other.

1.17 RULE CASCADING

- A single style sheet associated with one or more web pages is valuable, but in quite a limited way.
- For small sites, the single style sheet is sufficient, but for larger sites, especially sites managed by more than one person (perhaps several teams who may never communicate) single style sheets don't provide the ability to share common styles, and extend these styles where necessary. This can be a significant limitation.
- Cascading style sheets are unlike the style sheets you might have worked with using word processors, because they can be linked together to create a hierarchy of related style sheets.

Font Family

- The font family of a text is set with the font-family property. The font-family property should hold several font names as a "fallback" system.
- If the browser does not support the first font, it tries the next font. Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Example

```
p{font-family:"Times New Roman", Times, serif;}
```

Font Style

The font-style property is mostly used to specify italic text. This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Font Size

- The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs. Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
- The font-size value can be an absolute, or relative size.
 - Absolute size: Sets the text to a specified size. Does not allow a user to change the text size in all browsers (bad for accessibility reasons). Absolute size is useful when the physical size of the output is known
 - Relative size: Sets the size relative to surrounding elements. Allows a user to change the text size in browsers

The CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

- The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.
- The box model allows us to place a border around elements and space elements in relation to other elements.

Explanation of the different parts:

- **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent.
- **Border** - A border that goes around the padding and content. The border is affected by the background color of the box.
- **Padding** - Clears an area around the content. The padding is affected by the background color of the box .
- **Content** - The content of the box, where text and images appear.
- In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

1.18 INHERITANCE AND CASCADE

Inheritance and the cascade are two fundamental concepts in CSS, that are important to understand. The two concepts are closely related, yet different:

- Inheritance is associated with how the elements in the HTML markup inherit properties from their parent (containing) elements and pass them on to their children.
- The cascade relates to CSS declarations being applied to a document, and how conflicting rules do or do not override each other.

Inheritance

- Inheritance is the mechanism by which certain properties are passed on from a parent element down to its children, in the same fashion as genetics: if parents have blue eyes, their children will probably also have blue eyes.
- Not all CSS properties are inherited, because it does not make sense for some of them to be.

Why inheritance is useful

- CSS has an inheritance mechanism because otherwise CSS rules would be redundant. Without inheritance, it would be necessary to specify styles like font

family, font size, and text color individually — for every single element type. The code would become bloated and repetitive.

- Using inheritance, you can specify the font properties for the html or body elements and the styles will be inherited by all other elements. You can specify background and text colors for a specific container element and the text color will automatically be inherited by any child elements in that container.
- The background color is not inherited, but the initial value for background-color is transparent, which means a parent's background will shine through. The effect is similar to the page's appearance if background colors were inherited.

How inheritance works

- Every element in an HTML document inherits all inheritable properties from its parent except the root element (<html>), which does not have a parent.
- Whether or not the inherited properties will have any effect depends on other things, as described later in the section about the cascade.
- Just as a blue-eyed mother can have a brown-eyed child if the father has brown eyes, inherited properties in CSS can be overridden.
- **An example of inheritance** Copy the following HTML document into a new file in your favorite text editor and save it as inherit.html.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Inheritance</title>
  </head>
  <body>
    <h1>Heading</h1>
    <p>A paragraph of text.</p>
  </body>
</html>
```

Create a new empty file in your text editor, copy the following CSS rule into it, and save the file as style.css in the same location as the HTML file.

```
html {
  font: 75% Verdana, sans-serif;
}
```

Link the style sheet to your HTML document by inserting the following line before the </head> tag:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Save the modified HTML file and reload the document in your browser.

The CSS rule only specified two properties — the font size and the font family — but that rule is equivalent to the following:

```
html {
  font-style: normal;
  font-variant: normal;
  font-weight: normal;
  font-size: 75%;
  line-height: normal;
  font-family: Verdana, sans-serif;
}
```

1.19 BACKGROUNDS

CSS Background

CSS background properties are used to define the background effects of an element.

CSS properties used for background effects:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

1.19.1 Background Color

The background-color property specifies the background color of an element. The background color of a page is defined in the body selector:

Example

```
body {background-color:#b0c4de;}
```

The background color can be specified by:

name - a color name, like "red"

RGB - an RGB value, like
"rgb(255,0,0)" Hex - a hex value, like
"#ff0000"

1.19.2 Background Image

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element. The background image for a page can be set like this:

Example

```
body {background-image:url('paper.gif');}
```

1.19.3 Background Image - Repeat Horizontally or Vertically

By default, the background-image property repeats an image both horizontally and vertically. Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Example

```
body
{
background-image:url('gradient2.png');
```

- If the image is repeated only horizontally (repeat-x), the background will look better:

Example

```
body
{
background-image:url('gradient2.png'); background-repeat:repeat-x;
}
```

1.19.4 CSS Multiple Backgrounds

- CSS allows you to add multiple background images for an element, through the background-image property.
- The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.
- The following example has two background images, the first image is a flower (aligned to the bottom and right) and the second image is a paper background (aligned to the top-left corner):

Example

```
#example1 {
    background-image: url(img_flwr.gif), url(paper.gif);
    background-position: right bottom, left top;
    background-repeat: no-repeat, repeat;
}
```

- Multiple background images can be specified using either the individual background properties (as above) or the background shorthand property.

1.19.5 CSS Background Size

- The CSS background-size property allows you to specify the size of background images.
- The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.
- The following example resizes a background image to much smaller than the original image (using pixels):

Example

```
#div1 {
    background: url(img_flower.jpg);
    background-size: 100px 80px;
    background-repeat: no-repeat;
}
```

- The two other possible values for background-size are contain and cover.
- The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). As such, depending on the proportions of the background image and the background positioning area, there may be some areas of the background which are not covered by the background image.
- The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area). As such, some parts of the background image may not be visible in the background positioning area.
- The following example illustrates the use of contain and cover:

Example

```
#div1 {
    background: url(img_flower.jpg);
    background-size: contain;
    background-repeat: no-repeat;
}

#div2 {
    background: url(img_flower.jpg);
    background-size: cover;
    background-repeat: no-repeat;
}
```

1.20 BACKGROUND IMAGES

- The background-size property also accepts multiple values for background size (using a comma-separated list), when working with multiple backgrounds.
- The following example has three background images specified, with different background-size value for each image:

Example

```
#example1 {  
    background: url(img_tree.gif) left top no-repeat, url(img_flwr.gif) right bottom no-repeat,  
    url(paper.gif) left top repeat;  
    background-size: 50px, 130px, auto;  
}
```

Full Size Background Image

- Now we want to have a background image on a website that covers the entire browser window at all times.
- The requirements are as follows:
 - Fill the entire page with the image (no white space)
 - Scale image as needed
 - Center image on page
 - Do not cause scrollbars
 - The following example shows how to do it; Use the <html> element (the <html> element is always at least the height of the browser window). Then set a fixed and centered background on it. Then adjust its size with the background-size property:

Example

```
html {  
    background: url(img_man.jpg) no-repeat center fixed;  
    background-size: cover;  
}
```

CSS background-origin Property

- The CSS background-origin property specifies where the background image is positioned.
- The property takes three different values:
 - border-box - the background image starts from the upper left corner of the border
 - padding-box - (default) the background image starts from the upper left corner of the padding edge
 - content-box - the background image starts from the upper left corner of the content
- The following example illustrates the background-origin property:

Example

```
#example1 {  
    border: 10px solid black;  
    padding: 35px;  
    background: url(img_flwr.gif);  
    background-repeat: no-repeat;  
    background-origin: content-box;  
}
```

CSS background-clip Property

- The CSS background-clip property specifies the painting area of the background.
- The property takes three different values:
 - border-box - (default) the background is painted to the outside edge of the border
 - padding-box - the background is painted to the outside edge of the padding
 - content-box - the background is painted within the content box

- The following example illustrates the background-clip property:

Example

```
#example1 {
    border: 10px dotted black;
    padding: 35px;
    background: yellow;
    background-clip: content-box;
}
```

1.21 CSS COLORS

CSS supports 140+ color names, HEX values, RGB values, RGBA values, HSL values, HSLA values, and opacity.

1.21.1 RGBA Colors

- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
- An RGBA color value is specified with: `rgba(red, green, blue, alpha)`. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
rgba(255, 0, 0, 0.2);
rgba(255, 0, 0, 0.4);
rgba(255, 0, 0, 0.6);
rgba(255, 0, 0, 0.8);
```

The following example defines different RGBA colors:

Example

```
#p1 {background-color: rgba(255, 0, 0, 0.3);} /* red with opacity */
#p2 {background-color: rgba(0, 255, 0, 0.3);} /* green with opacity */
#p3 {background-color: rgba(0, 0, 255, 0.3);} /* blue with opacity */
```

1.21.2 HSL Colors

- HSL stands for Hue, Saturation and Lightness.
- An HSL color value is specified with: `hsl(hue, saturation, lightness)`.
- Hue is a degree on the color wheel (from 0 to 360):
 - 0 (or 360) is red
 - 120 is green
 - 240 is blue
- Saturation is a percentage value: 100% is the full color.
- Lightness is also a percentage; 0% is dark (black) and 100% is white.

```
hsl(0, 100%, 30%);
hsl(0, 100%, 50%);
hsl(0, 100%, 70%);
hsl(0, 100%, 90%);
```

The following example defines different HSL colors:

Example

```
#p1 {background-color: hsl(120, 100%, 50%);} /* green */
#p2 {background-color: hsl(120, 100%, 75%);} /* light green */
#p3 {background-color: hsl(120, 100%, 25%);} /* dark green */
#p4 {background-color: hsl(120, 60%, 70%);} /* pastel green */
```

1.21.3 HSLA Colors

- HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

- An HSLA color value is specified with: hsla(hue, saturation, lightness, alpha), where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
hsla(0, 100%, 30%, 0.3);
hsla(0, 100%, 50%, 0.3);
hsla(0, 100%, 70%, 0.3);
hsla(0, 100%, 90%, 0.3);
```

- The following example defines different HSLA colors:

Example

```
#p1 {background-color: hsla(120, 100%, 50%, 0.3);} /* green with opacity */
#p2 {background-color: hsla(120, 100%, 75%, 0.3);} /* light green with opacity */
#p3 {background-color: hsla(120, 100%, 25%, 0.3);} /* dark green with opacity */
#p4 {background-color: hsla(120, 60%, 70%, 0.3);} /* pastel green with opacity */
```

1.21.4 Opacity

- The CSS opacity property sets the opacity for the whole element (both background color and text will be opaque/transparent).
- The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Example

```
#p1 {background-color:rgb(255,0,0);opacity:0.6;} /* red with opacity */
#p2 {background-color:rgb(0,255,0);opacity:0.6;} /* green with opacity */
#p3 {background-color:rgb(0,0,255);opacity:0.6;} /* blue with opacity */
```

1.22 CSS SHADOW EFFECTS

CSS Shadow Effects

With CSS you can add shadow to text and to elements.

- text-shadow
- box-shadow

1.22.1 CSS Text Shadow

- The CSS text-shadow property applies shadow to text.
- In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

Text shadow effect!

Example

```
h1 {
  text-shadow: 2px 2px;
}
```

Text shadow effect!

Example

```
h1 {
  text-shadow: 2px 2px red;
}
```

Multiple Shadows

- To add more than one shadow to the text, you can add a comma-separated list of shadows.
- The following example shows a red and blue neon glow shadow:

Text shadow effect!

Example

```
h1 {  
    text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;  
}
```

CSS box-shadow Property

- The CSS box-shadow property applies shadow to elements.
- In its simplest use, you only specify the horizontal shadow and the vertical shadow:
- This is a yellow <div> element with a black box-shadow

Example

```
div {  
    box-shadow: 10px 10px;  
}
```

- Next, add a color to the shadow:
- This is a yellow <div> element with a grey box-shadow

Example

```
div {  
    box-shadow: 10px 10px grey;  
}
```

- Next, add a blur effect to the shadow:
- This is a yellow <div> element with a blurred, grey box-shadow

Example

```
div {  
    box-shadow: 10px 10px 5px grey;  
}
```

Example

```
#boxshadow {  
    position: relative;  
    box-shadow: 1px 2px 4px rgba(0, 0, 0, .5);  
    padding: 10px;  
    background: white;  
}  
  
#boxshadow img {  
    width: 100%;  
    border: 1px solid #8a4419;  
    border-style: inset;  
}  
  
#boxshadow::after {  
    content: "";  
    position: absolute;  
    z-index: -1; /* hide shadow behind image */  
    box-shadow: 0 15px 20px rgba(0, 0, 0, 0.3);  
    width: 70%;  
    left: 15%; /* one half of the remaining 30% */  
    height: 100px;  
    bottom: 0;  
}
```

1.23 CSS TEXT EFFECTS

CSS Text Overflow, Word Wrap, Line Breaking Rules, and Writing Modes

- text-overflow

- word-wrap
- word-break
- writing-mode

1.23.1 CSS Text Overflow

- The CSS text-overflow property specifies how overflowed content that is not displayed should be signaled to the user. It can be clipped:
- This is some long text that will not fit in the box or it can be rendered as an ellipsis (...): This is some long text that will not fit in the box. The CSS code is as follows:

Example

```
p.test1 {
    white-space: nowrap;
    width: 200px;
    border: 1px solid #000000;
    overflow: hidden;
    text-overflow: clip;
}

p.test2 {
    white-space: nowrap;
    width: 200px;
    border: 1px solid #000000;
    overflow: hidden;
    text-overflow: ellipsis;
}
```

The following example shows how you can display the overflowed content when hovering over the element:

Example

```
div.test:hover {
    overflow: visible;
}
```

CSS Word Wrapping

- The CSS word-wrap property allows long words to be able to be broken and wrap onto the next line.
- If a word is too long to fit within an area, it expands outside:
- This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.
- The word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word:
- This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

The CSS code is as follows:

Example

Allow long words to be able to be broken and wrap onto the next line:

```
p {
    word-wrap: break-word;
}
```

CSS Word Breaking

- The CSS word-break property specifies line breaking rules.
- This paragraph contains some text. This line will-break-at-hyphens.
- This paragraph contains some text. The lines will break at any character.

The CSS code is as follows:

Example

```
p.test1 {  
    word-break: keep-all;  
}
```

```
p.test2 {  
    word-break: break-all;  
}
```

CSS Writing Mode

- The CSS writing-mode property specifies whether lines of text are laid out horizontally or vertically.
- Some text with a span element with a vertical-rl writing-mode.
- The following example shows some different writing modes:

Example

```
p.test1 {  
    writing-mode: horizontal-tb;  
}
```

```
span.test2 {  
    writing-mode: vertical-rl;  
}
```

```
p.test2 {  
    writing-mode: vertical-rl;  
}
```

CSS Text Effect Properties

The following table lists the CSS text effect properties:

Property	Description
<u>text-align-last</u>	Specifies how to align the last line of a text
<u>text-justify</u>	Specifies how justified text should be aligned and spaced
<u>text-overflow</u>	Specifies how overflowed content that is not displayed should be signaled to the user
<u>word-break</u>	Specifies line breaking rules for non-CJK scripts
<u>word-wrap</u>	Allows long words to be able to be broken and wrap onto the next line
<u>writing-mode</u>	Specifies whether lines of text are laid out horizontally or vertically

1.24 CSS 2D TRANSFORMS

- CSS transforms allow you to move, rotate, scale, and skew elements. Mouse over the element below to see a 2D transformation:

CSS 2D Transforms Methods

With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()

- skewX()
- skewY()
- skew()
- matrix()

1.24.1 The translate() Method

- The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).
- The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position:

Example

```
div {
  transform: translate(50px, 100px);
}
```

1.24.2 The rotate() Method

- The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.
- The following example rotates the <div> element clockwise with 20 degrees:



Example

```
div {
  transform: rotate(20deg);
}
```

Using negative values will rotate the element counter-clockwise.

The following example rotates the <div> element counter-clockwise with 20 degrees:

Example

```
div {
  transform: rotate(-20deg);
}
```

1.24.3 The scale() Method

- The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).
- The following example increases the <div> element to be two times of its original width, and three times of its original height:

Example

```
div {
  transform: scale(2, 3);
}
```

The following example decreases the <div> element to be half of its original width and height:



Example

```
div {
  transform: scale(0.5, 0.5);
}
```

1.24.4 The scaleX() Method

- The scaleX() method increases or decreases the width of an element.
- The following example increases the <div> element to be two times of its original width:

Example

```
div {  
    transform: scaleX(2);  
}
```

The following example decreases the <div> element to be half of its original width:

Example

```
div {  
    transform: scaleX(0.5);  
}
```

1.24.5 The scaleY() Method

- The scaleY() method increases or decreases the height of an element.
- The following example increases the <div> element to be three times of its original height:

Example

```
div {  
    transform: scaleY(3);  
}
```

The following example decreases the <div> element to be half of its original height:

Example

```
div {  
    transform: scaleY(0.5);  
}
```

1.24.6 The skewX() Method

- The skewX() method skews an element along the X-axis by the given angle.
- The following example skews the <div> element 20 degrees along the X-axis:

Example

```
div {  
    transform: skewX(20deg);  
}
```

1.24.7 The skewY() Method

- The skewY() method skews an element along the Y-axis by the given angle.
- The following example skews the <div> element 20 degrees along the Y-axis:

Example

```
div {  
    transform: skewY(20deg);  
}
```

1.24.8 The skew() Method

- The skew() method skews an element along the X and Y-axis by the given angles.
- The following example skews the <div> element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

Example

```
div {  
    transform: skew(20deg, 10deg);  
}
```

- If the second parameter is not specified, it has a zero value. So, the following example skews the <div> element 20 degrees along the X-axis:

Example

```

div {
  transform: skew(20deg);
}

```

1.24.9 The matrix() Method

- The matrix() method combines all the 2D transform methods into one.
- The matrix() method takes six parameters, containing mathematical functions, which allows you to rotate, scale, move (translate), and skew elements.
- The parameters are as follows:
matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

Example

```

div {
  transform: matrix(1, -0.3, 0, 1, 0, 0);
}

```

CSS Transform Properties

The following table lists all the 2D transform properties:

Property	Description
<u>transform</u>	Applies a 2D or 3D transformation to an element
<u>transform-origin</u>	Allows you to change the position on transformed elements

CSS 2D Transform Methods

Function	Description
matrix(n,n,n,n,n,n)	Defines a 2D transformation, using a matrix of six values
translate(x,y)	Defines a 2D translation, moving the element along the X- and the Y-axis
translateX(n)	Defines a 2D translation, moving the element along the X-axis
translateY(n)	Defines a 2D translation, moving the element along the Y-axis
scale(x,y)	Defines a 2D scale transformation, changing the element's width and height
scaleX(n)	Defines a 2D scale transformation, changing the element's width
scaleY(n)	Defines a 2D scale transformation, changing the element's height
rotate(angle)	Defines a 2D rotation, the angle is specified in the parameter

Defines a 2D skew transformation along the X- and the Y-



skew(x-angle,y-angle)	Defines a 2D skew transformation along the X- and the Y-axis
skewX(angle)	Defines a 2D skew transformation along the X-axis
skewY(angle)	Defines a 2D skew transformation along the Y-axis

1.25 CSS TRANSITIONS

- CSS transitions allow you to change property values smoothly, over a given duration.
- Mouse over the element below to see a CSS transition effect:

CSS

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

Browser Support for Transitions

The numbers in the table specify the first browser version that fully supports the property.

Property

transition	26.0	10.0	16.0	6.1	12.1
transition-delay	26.0	10.0	16.0	6.1	12.1
transition-duration	26.0	10.0	16.0	6.1	12.1
transition-property	26.0	10.0	16.0	6.1	12.1
transition-timing-function	26.0	10.0	16.0	6.1	12.1

Browser Specific Prefixes

Some older browsers need specific prefixes (-webkit-) to understand the transition properties:

Example

```
div {
  width: 100px;
  height: 100px;
  background: red;
  -webkit-transition: width 2s; /* Safari prior 6.1 */
  transition: width 2s;
}
```

How to Use CSS Transitions?

To create a transition effect, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

The following example shows a 100px * 100px red <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds:

Example

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
```

- The transition effect will start when the specified CSS property (width) changes value.

Now, let us specify a new value for the width property when a user mouses over the <div> element:

Example

```
div:hover {
  width: 300px;
}
```

Notice that when the cursor mouses out of the element, it will gradually change back to its original style.

Change Several Property Values

The following example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height:

Example

```
div {  
    transition: width 2s, height 4s;  
}
```

Specify the Speed Curve of the Transition

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end
- cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

The following example shows some of the different speed curves that can be used:

Example

```
#div1 {transition-timing-function: linear;}  
#div2 {transition-timing-function: ease;}  
#div3 {transition-timing-function: ease-in;}  
#div4 {transition-timing-function: ease-out;}  
#div5 {transition-timing-function: ease-in-out;}
```

Delay the Transition Effect

The transition-delay property specifies a delay (in seconds) for the transition effect.

The following example has a 1 second delay before starting:

Example

```
div {  
    transition-delay: 1s;  
}
```

Transition + Transformation

The following example adds a transition effect to the transformation:

Example

```
div {  
    transition: width 2s, height 2s, transform 2s;  
}
```

1.26 CSS ANIMATIONS

- CSS allows animation of HTML elements without using JavaScript or Flash!

CSS

- @keyframes
- animation-name
- animation-duration

- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

Browser Support for Animations

The numbers in the table specify the first browser version that fully supports the property.

Property

@keyframes	43.0	10.0	16.0	9.0	30.0
animation-name	43.0	10.0	16.0	9.0	30.0
animation-duration	43.0	10.0	16.0	9.0	30.0
animation-delay	43.0	10.0	16.0	9.0	30.0
animation-iteration-count	43.0	10.0	16.0	9.0	30.0
animation-direction	43.0	10.0	16.0	9.0	30.0
animation-timing-function	43.0	10.0	16.0	9.0	30.0
animation-fill-mode	43.0	10.0	16.0	9.0	30.0
animation	43.0	10.0	16.0	9.0	30.0

What are CSS Animations?

- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.

1.26.1 The @keyframes Rule

- When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.
- To get an animation to work, you must bind the animation to an element.
- The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

Example

```
/* The animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
```

```
    animation-duration: 4s;  
}
```

- **Note:** The animation-duration property defines how long time an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).
- In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).
- It is also possible to use percent. By using percent, you can add as many style changes as you like.
- The following example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

Example

```
/* The animation code */  
@keyframes example {  
    0% {background-color: red;}  
    25% {background-color: yellow;}  
    50% {background-color: blue;}  
    100% {background-color: green;}  
}  
  
/* The element to apply the animation to */  
div {  
    width: 100px;  
    height: 100px;  
    background-color: red;  
    animation-name: example;  
    animation-duration: 4s;  
}
```

- The following example will change both the background-color and the position of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	Description
<u>@keyframes</u>	Specifies the animation code
<u>animation</u>	A shorthand property for setting all the animation properties
<u>animation-delay</u>	Specifies a delay for the start of an animation
<u>animation-direction</u>	Specifies whether an animation should be played forwards, backwards or in alternate cycles
<u>animation-duration</u>	Specifies how long time an animation should take to complete one cycle
<u>animation-fill-mode</u>	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
<u>animation-iteration-count</u>	Specifies the number of times an animation should be played
<u>animation-name</u>	Specifies the name of the @keyframes animation
<u>animation-play-state</u>	Specifies whether the animation is running or paused

animation-timing-function Specifies the speed curve of the animation