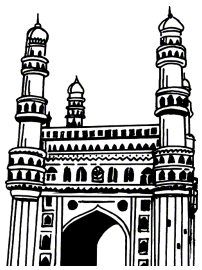
Rahul's ✔
Topper's Voice

AS PER
CBCS SYLLABUS



B.Sc.

III Year VI Sem

Latest 2022 Edition

WEB TECHNOLOGIES

- ☞ **Study Manual**
- ☞ **Short Question Answers**
- ☞ **Important Questions**
- ☞ **Lab Practicals**
- ☞ **Choose the Correct Answers**
- ☞ **Fill in the blanks**
- ☞ **Solved Model Papers**

-.249/-

- by -

WELL EXPERIENCED LECTURER

Rahul Publications™
Hyderabad. Ph : 66550071, 9391018098

B.Sc.

III Year VI Sem

WEB TECHNOLOGIES

Inspite of many efforts taken to present this book without errors, some errors might have crept in. Therefore we do not take any legal responsibility for such errors and omissions. However, if they are brought to our notice, they will be corrected in the next edition.

Price ` 249-00

Sole Distributors : ☎ : **66550071, Cell : 9391018098**

VASU BOOK CENTRE

Shop No. 2, Beside Gokul Chat, Koti, Hyderabad.

Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.

Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.

WEB TECHNOLOGIES

CONTENTS

STUDY MANUAL

Important Questions	**V - VIII**
Unit - I	**1 - 78**
Unit - II	**75 - 120**
Unit - III	**121 - 168**
Unit - IV	**169 - 224**
Lab Practicals	**225 - 272**

SOLVED MODEL PAPERS

Model Paper - I	**273 - 273**
Model Paper - II	**274 - 274**
Model Paper - III	**275 - 275**

SYLLABUS

UNIT - I

Introduction To XHTML– Introduction, first HTML, Headings, Linking, Images, special characters and horizontal rules, Lists, Tables, Frames, Forms, internal linking, meta Elements. CASCADING STYLE SHEETS – Introduction, Inline Styles, Embedded Style Sheets, Conflicting Styles, Linking external sheets, position Elements, box model and text flow, media types, building a CSS drop-down menu, user style sheets, CSS3.

UNIT - II

Introduction To Java Scripting- introduction, simple program, prompt dialog and alert boxes, memory concepts, operators, decision making, control structures, if… else statement, while, counter-controlled repetitions, switch statement, do… while statement, break and continue statements. Functions – program modules in JavaScript, programmer–defined functions, functions definition, scope rules, global functions, Recursion.

UNIT - III

Arrays - Introduction, Declaring and Allocating Arrays, References and Reference Parameters, Passing Arrays to Functions. Multidimensional Arrays,

EVENTS – Registering Event Handling, Event Onload, Onmouseover, Onmouseout, Onfocus, Onblur, Onsubmit, Onreset, Event Bubbling, More Events.

JAVA SCRIPT OBJECTS – Introduction to Object Technology, Math Object, String Object, Date Object, Boolean and Number Object, Document and Window Objects, Using Cookies.

UNIT - IV

XML - Introduction, XML Basics, Structuring Data, XML Namespaces, Document Type Definitions (DTDs), W3C XML Schema Documents, XML Vocabularies, Extensible Style sheet Language and XSL Transformations, Document Object Model (DOM).

Ajax-Enabled Rich Internet Applications: introduction, history of Ajax, traditional web applications Vs Ajax Applications, RIAs with Ajax, Ajax example using XML HttpRequest object, XML and DOM, creating full scale Ajax-enabled application, Dojo Toolkit.

Contents

Topic **Page No.**

UNIT - I

1.1 Introduction to Xhtml 1
1.1.1 Introduction 1
1.1.2 First HTML 1
1.1.3 Headings 2
1.1.4 Linking 3
1.1.5 Images 4
1.1.6 Special characters and horizontal rules 6
1.1.7 Lists 8
1.1.8 Tables 15
1.1.9 Frames 20
1.1.10 Forms 23
1.1.11 Internal linking 37
1.1.12 Meta Elements 39
1.2 Cascading Style Sheets 40
1.2.1 Introduction 40
1.2.2 Inline styles 46
1.2.3 Embedded Style Sheets 47
1.2.4 Conflicting Styles 48
1.2.5 Linking External Sheets 50
1.2.6 Position Elements 50
1.2.7 Box Model And Text Flow 53
1.2.8 Media Types 59
1.2.9 Building A Css Drop-down Menu 61
1.2.10 User Style Sheets 64
1.2.11 CSS3 66

- **Short Questions Answer ..... 67 - 71**
- **Choose the Correct Answer ..... 72 - 73**
- **Fill in the blanks ..... 74 - 74**

Topic **Page No.**

UNIT - II

2.1 Introduction to Java Scripting 75

2.1.1 Introduction 75

2.1.2 Simple Program 76

2.1.3 Prompt Dialog and Alert Boxes 77

2.1.4 Memory Concepts 81

2.1.5 Operators 83

2.1.6 Decision Making 86

2.1.7 Control Structures 89

2.1.8 If… Else Statement 90

2.1.9 While 92

2.1.10 Counter-controlled Repetitions 92

2.1.11 Switch Statement 93

2.1.12 Do… While Statement 94

2.1.13 Break And Continue Statements 95

2.2 Functions 97

2.2.1 Program Modules In Javascript 97

2.2.2 Programmer–Defined Functions 98

2.2.3 Functions Definition 104

2.2.4 Scope Rules 106

2.2.5 Global Functions108 108

2.2.6 Recursion 108

- **Short Questions Answer ..... 110 - 117**
- **Choose the Correct Answer ..... 118 - 119**
- **Fill in the blanks ..... 120 - 120**

UNIT - III

3.1 Arrays 121

3.1.1 Introduction 121

3.1.2 Declaring and Allocating Arrays 121

3.1.3 References and Reference Para-meters 123

Topic **Page No.**

3.1.4 Passing Arrays to Functions 124
3.1.5 Multidimensional Arrays 125
3.2 Events 127
3.2.1 Registering Event Handling 127
3.2.2 Event Onload 129
3.2.3 On Mouseover 130
3.2.4 On Mouseout 130
3.2.5 Onfocus 131
3.2.6 ONBLUR 131
3.2.7 ONSUBMIT 133
3.2.8 ONRESET 134
3.2.9 Event Bubbling 134
3.2.10 MORE EVENTS 137
3.3 Java Script Objects 142
3.3.1 Introduction to Object Technology 142
3.3.2 Math Object 144
3.3.3 String Object 147
3.3.4 Date Object 150
3.3.5 Boolean and Number Object 153
3.3.6 Document and Window Objects 155
3.3.7 Using Cookies 159

- **Short Questions Answer ..... 161 - 166**
- **Choose the Correct Answer ..... 167 - 167**
- **Fill in the blanks ..... 168 - 168**

UNIT - IV

4.1 XML 169
4.1.1 Introduction 169
4.1.2 XML Basics 169
4.1.3 Structuring Data 174
4.1.4 XML Namespaces 175
4.1.5 Document Type Definitions (DTDS) 178
4.1.6 W3C XML Schema Documents 183
4.1.7 XML Vocabularies 185

Topic **Page No.**

4.1.8 Extensible Style Sheet Language and XSL Transformations 186
4.1.9 Document Object Model (DOM) 195
4.2 Ajax-enabled Rich Internet Applications 199
4.2.1 Introduction 199
4.2.2 History of Ajax 200
4.2.3 Traditional Web Applications Vs Ajax Applications 200
4.2.4 Rias with Ajax 201
4.2.5 Ajax Example Using XMLHTTPRequest Object 203
4.2.6 XML AND DOM 205
4.2.7 Creating full Scale Ajax - Enabled Application 208
4.2.8 DOJO TOOLKIT 211

- **Short Questions Answer ..... 213 - 221**
- **Choose the Correct Answer ..... 222 - 223**
- **Fill in the blanks ..... 224 - 224**

Important Questions

UNIT - I

1. Explain the functionality of <anchor> tag.

Ans:

Refer Unit-I, Q.No. 4.

2. Write about the special characers/ character entieies in HTML.

Ans :

Refer Unit-I, Q.No. 6.

3. Explain various types of lists used in HTML.

Ans:

Refer Unit-I, Q.No. 8.

4. Explain about HTML <table> tag with an example.

Ans:

Refer Unit-I, Q.No. 9.

5. Explain briefly about <form> tags.

Ans:

Refer Unit-I, Q.No. 11.

6. Explain with an example how to inter link the document in HTML.

Ans :

Refer Unit-I, Q.No. 13.

7. What is CSS? Explain, how CSS can be used in HTML to change the style of the document.

Ans:

Refer Unit-I, Q.No. 15.

8. How CSS will avoid overridden? Explain.

Ans:

Refer Unit-I, Q.No. 18.

9. Explain CSS media rule property with example.

Ans:

Refer Unit-I, Q.No. 23.

UNIT - II

1. How to use prompt() dialog box in Java Script? Explain.

Ans:

Refer Unit-II, Q.No. 3.

2. Explain about the memory allocation concept in Java Script.

Ans:

Refer Unit-II, Q.No. 5.

3. Explain various arithemetic operators used in JavaScript with examples.

Ans:

Refer Unit-II, Q.No. 6.

4. Explain about control structures in Java Script.

Ans:

Refer Unit-II, Q.No. 8.

5. Explain Java script Export and Import Modules.

Ans:

Refer Unit-II, Q.No. 15.

6. What is function in javascript? Explain about user defined functions.

Ans:

Refer Unit-II, Q.No. 16.

7. Explain how to define and use a function in Java script.

Ans:

Refer Unit-II, Q.No. 17.

UNIT - III

1. Explain how to create and use array in Javascript.

Ans :

Refer Unit-III, Q.No. 2.

2. How Do you reference The Elements In An Array?

Ans :

Refer Unit-III, Q.No. 3.

3. Explain how to create and access the multi dimensional arrays in Javascript.

Ans :

Refer Unit-III, Q.No. 5.

4. Write briefly about event handling in java script.

Ans :

Refer Unit-III, Q.No. 6.

5. Explain onload event in javascript with example.

Ans :

Refer Unit-III, Q.No. 7.

6. Explain onblur event in javascript with example.

Ans :

Refer Unit-III, Q.No. 11.

7. Explain onsubmit event in javascript with example.

Ans :

Refer Unit-III, Q.No. 12.

8. What is event bubbling in java script? Explain with an example.

Ans :

Refer Unit-III, Q.No. 14.

9. Explain Javascript onclick event.

Ans :

Refer Unit-III, Q.No. 15.

10. Explain JavaScript onresize event.

Ans :

Refer Unit-III, Q.No. 17.

11. Explain about Javascript objects with example.

Ans :

Refer Unit-III, Q.No. 18.

12. Explain about math object in javascript.

Ans :

Refer Unit-III, Q.No. 19.

13. Write about Document Object Model.

Ans :

Refer Unit-III, Q.No. 23.

UNIT - IV

1. Write the features of XML

Ans:

Refer Unit-IV, Q.No. 2.

2. Explain XML DTD with an example.

Ans :

Refer Unit-IV, Q.No. 7.

3. What is XSL? Explain about it.

Ans:

Refer Unit-IV, Q.No. 10.

4. What is XML DOM? Explain it with example.

Ans:

Refer Unit-IV, Q.No. 11.

5. What is Ajax? What are the various technologies which supports AJAX.

Ans:

Refer Unit-IV, Q.No. 12.

6. Write about the importance of Rich Internet Applications with AJAX.

Ans:

Refer Unit-IV, Q.No. 15.

7. What is XMLHttpRequest Object? How to create an XMLHttpRequest Object.

Ans:

Refer Unit-IV, Q.No. 16.

8. How AXAX can be communicated with XML file? Explain.

Ans:

Refer Unit-IV, Q.No. 17.

9. Explain, how to create full scale AJAX application.

Ans:

Refer Unit-IV, Q.No. 19,

UNIT I

Introduction To XHTML– Introduction, first HTML, Headings, Linking, Images, special characters and horizontal rules, Lists, Tables, Frames, Forms, internal linking, meta Elements. CASCADING STYLE SHEETS – Introduction, Inline Styles, Embedded Style Sheets, Conflicting Styles, Linking external sheets, position Elements, box model and text flow, media types, building a CSS drop-down menu, user style sheets, CSS3.

1.1 Introduction to XHTML

1.1.1 Introduction

Q1. What is HTML? Give brief description about that.

Ans :

HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications.

- **Hyper Text:** HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.
- **Markup language:** A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.
- **Web Page:** A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages.

Features of HTML

- It is easy to learn and easy to use.
- It is platform-independent.
- Images, videos, and audio can be added to a web page.
- Hypertext can be added to the text.
- It is a markup language.

Why learn HTML?

- It is a simple markup language. Its implementation is easy.
- It is used to create a website.
- Helps in developing fundamentals about web programming.
- Boost professional career.

Advantages

- HTML is used to build websites.
- It is supported by all browsers.
- It can be integrated with other languages like CSS, JavaScript, etc.

Disadvantages

- HTML can only create static web pages. For dynamic web pages, other languages have to be used.
- A large amount of code has to be written to create a simple web page.
- The security feature is not good.

1.1.2 First HTML

Q2. Describe HTML page structure.

Ans :

HTML Page Structure

Below is a visualization of an HTML page structure:

```
<html>
<head>
<title>Page title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

Description of HTML Example

<!DOCTYPE>: It defines the document type or it instruct the browser about the version of HTML.

<html > :This tag informs the browser that it is an HTML document. Text between html tag describes the web document. It is a container for all other elements of HTML except <!DOCTYPE>

<head>: It should be the first element inside the <html> element, which contains the metadata(information about the document). It must be closed before the body tag opens.

<title>: As its name suggested, it is used to add title of that HTML page which appears at the top of the browser window. It must be placed inside the head tag and should close immediately. (Optional)

<body> : Text between body tag describes the body content of the page that is visible to the end user. This tag contains the main content of the HTML document.

<h1> : Text between <h1> tag describes the first level heading of the webpa

<p> : Text between <p> tag describes the paragraph of the webpage.

1.1.3 Headings

Q3. Explain the types of heading used in HTML with an example.

Ans :

A HTML heading or HTML h tag can be defined as a title or a subtitle which you want to display on the webpage. When you place the text within the heading tags <h1>.....</h1>, it is displayed on the browser in the bold format and size of the text depends on the number of heading.

There are six different HTML headings which are defined with the <h1> to <h6> tags, from highest level h1 (main heading) to the least level h6 (least important heading).

h1 is the largest heading tag and h6 is the smallest one. So h1 is used for most important heading and h6 is used for least important.

Headings in HTML helps the search engine to understand and index the structure of web page.

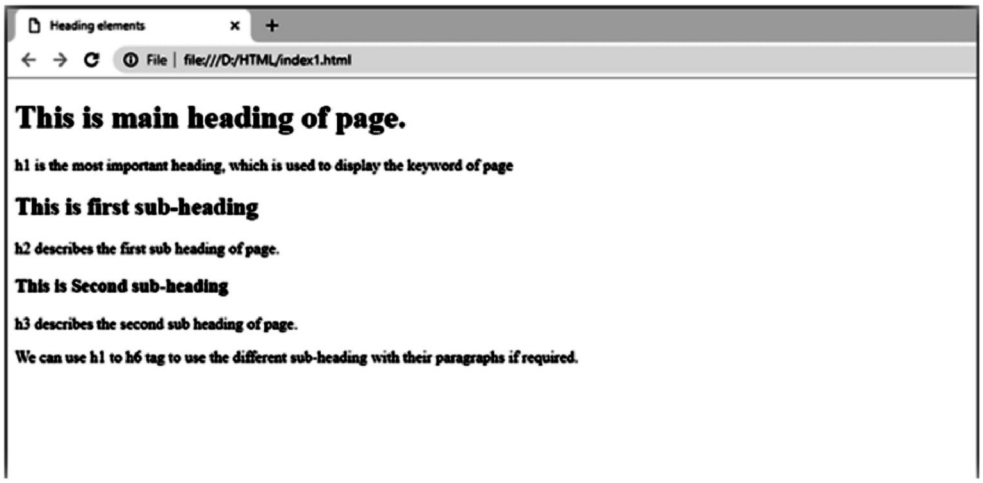
```
<h1>Heading no. 1</h1>
<h2>Heading no. 2</h2>
<h3>Heading no. 3</h3>
<h4>Heading no. 4</h4>
<h5>Heading no. 5</h5>
<h6>Heading no. 6</h6>
```

Example:

```
<!DOCTYPE html>
<html>  <head>
    <title>Heading elements</title>
 </head>
 <body>
     <h1>This is main heading of page. </
     h1>
```

```
<p>h1 is the most important heading, which is used to display the keyword of page
</ p>
<h2>This is first sub-heading</h2>
<p>h2 describes the first sub heading of page. </p>
<h3>This is Second sub-heading</h3>
<p>h3 describes the second sub heading of page.</p>
<p>We can use h1 to h6 tag to use the different sub-heading with their paragraphs if
required. </p>
</body> </html>
```

Output:



1.1.4 Linking

Q4. Explain how can you link the web pages using HTML

(OR)

Explain the functionality of <anchor> tag.

Ans : **(Imp.)**

The HTML <anchor> tag defines a hyperlink that links one page to another page. It can create hyperlink to other web page as well as files, location, or any URL. The "href" attribute is the most important attribute of the HTML a tag. and which links to destination page or URL.

href attribute of HTML anchor tag

The href attribute is used to define the address of the file to be linked. In other words, it points out the destination page.

The syntax of HTML anchor tag is given below.

```
<a href = ".....">  Link Text </a>
```

Let's see an example of HTML anchor tag.

```
<a href="second.html">Click for Second Page</a>
```

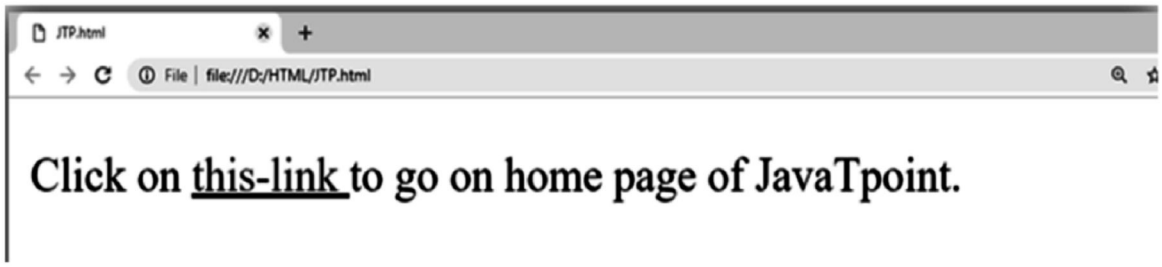
Specify a location for Link using target attribute

If we want to open that link to another page then we can use target attribute of <a> tag. With the **help of this link will be open in next page.**

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
<p>Click on <a href="https://www.javatpoint.com/" target="_blank"> this-link </a>to go on home page of JavaTpoint.</p>
</body>
</html>
```

Output:



Note:

- The **target** attribute can only use with href attribute in anchor tag.
- If we will not use target attribute then link will open in same page.

Appearance of HTML anchor tag

- An unvisited link is displayed underlined and blue.
- A visited link displayed underlined and purple.
- An active link is underlined and red.

1.1.5 Images

Q5. Explain the usage of <img> tag.

Ans :

HTML img tag is used to display image on the web page. HTML img tag is an empty tag that contains attributes only, closing tags are not used in HTML image element.

Let's see an example of HTML image.

```
<h2>HTML Image Example</h2>
<img src="good_morning.jpg" alt="Good Morning Friends"/>
```

Output:

Attributes of HTML img tag

The src and alt are important attributes of HTML img tag. All attributes of HTML image tag are given below.

1. **src :** It is a necessary attribute that describes the source or path of the image. It instructs the browser where to look for the image on the server.

   The location of image may be on the same directory or another server.

2. **alt :** The alt attribute defines an alternate text for the image, if it can't be displayed. The value of the alt attribute describe the image in words. The alt attribute is considered good for SEO prospective.

3. **width :** It is an optional attribute which is used to specify the width to display the image. It is not recommended now. You should apply CSS in place of width attribute.

4. **height :** It h3 the height of the image. The HTML height attribute also supports iframe, image and object elements. It is not recommended now. You should apply CSS in place of height attribute.

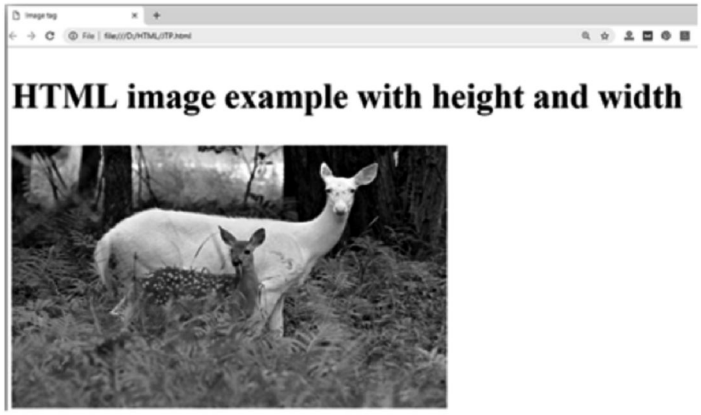
Use of height and width attribute with img tag

You have learnt about how to insert an image in your web page, now if we want to give some height and width to display image according to our requirement, then we can set it with height and width attributes of image.

Example:

```
<img src="animal.jpg" height="180" width="300" alt="animal image">
```

Output:



Use of alt attribute

We can use alt attribute with <img> tag. It will display an alternative text in case if image cannot be displayed on browser. Following is the example for alt attribute:

```
<img src="animal.png" height="180" width="300" alt="animal image">
```

Output:

How to get image from another directory/folder?

To insert an image in your web, that image must be present in your same folder where you have put the HTML file. But if in some case image is available in some other directory then you can access the image like this:

```
<img src="E:/images/animal.png" height="180" width="300" alt="animal image">
```

In above statement we have put image in local disk E———>images folder———>animal.png.

Use <img> tag as a link

We can also link an image with other page or we can use an image as a link. To do this, put <img> tag inside the <a> tag.

Example:

```
<a href="https://www.javatpoint.com/what-is-
robotics"><img src="robot.jpg" height="100" width="100"></a>
```


1.1.6 Special characters and horizontal rules

Q6. Write about the special characers/ character entieies in HTML.

Ans :

HTML character entities are used as a replacement of reserved characters in HTML. You can also replace characters that are not present on your keyboard by entities.

These characters are replaced because some characters are reserved in HTML. HTML entities provide a wide range of characters which can allow you to add icons, geometric shapes, mathematical operators, etc.

For example, if you use less than (<) or greater than (>) symbols in your text, the browser can mix them with tags that's why character entities are used in HTML to display reserved characters.

How to use an entity:

You can use an entity in your HTML document by name or by a numerical character reference. Each entity starts with symbol ampersand (&) and ends with a semicolon (;).

Syntax:

&entity_name;

OR

&#entity_number;

Most used HTML Character Entities

Result	Description	Entity Name	Entity Number
	non-breaking space		160
<	less than	<	60
>	greater than	>	62
&	ampersand	&	38
"	double quotation mark	"	34
'	single quotation mark (apostrophe)	'	39
¢	cent	¢	162
£	pound	£	163
¥	yen	¥	165
€	Euro	€	8364
©	copyright	©	169
®	registered trademark	®	174

Advantage of entity name: An entity name is easy to remember.

Disadvantage of entity name: Browsers may not support all entity names, but the support for numbers is good.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title></title>
</head>
 <body>
```

```
<h3>HTML entity example</h3>
<p> "This is the content written within entity"</p>
<p> <p> Paragraph tag </p>
</body>
</html>
```

Q7. Write about <hr> tag in HTML.

Ans :

HTML <hr> tag is used to specify a paragraph-level thematic break in HTML document. It is used when you abruptly change your topic in your HTML document. It draw a horizontal line between them. It is also called a Horizontal Rule in HTML.

```
HTML hr tag
<h2>HTML</h2>
<p>HTML is a language for describing web pages.</p>
<hr/>
<h2>HR Tag </h2>
<p> HR tag is used to draw a horizontal line within the texts to sepate content.<p>
```

Output:

```
HTML
HTML is a language for describing web pages.
HR Tag
HR tag is used to draw a horizontal line within the texts to separate content.
```

1.1.7 Lists

Q8. Explain various types of lists used in HTML.

Ans : **(Imp.)**

HTML Lists are used to specify lists of information. All lists may contain one or more list elements. There are three different types of HTML lists:

1. Ordered List or Numbered List (ol)
2. Unordered List or Bulleted List (ul)
3. Description List or Definition List (dl)

HTML Ordered List | HTML Numbered List

HTML Ordered List or Numbered List displays elements in numbered format. The HTML ol tag is used for ordered list. We can use ordered list to represent items either in numerical order format or alphabetical order format, or any format where an order is emphasized. There can be different types of numbered list:

- Numeric Number (1, 2, 3)
- Capital Roman Number (I II III)

- Small Romal Number (i ii iii)
- Capital Alphabet (A B C)
- Small Alphabet (a b c)

To represent different ordered lists, there are 5 types of attributes in <ol> tag.

Type	Description
Type "1"	This is the default type. In this type, the list items are numbered with numbers.
Type "I"	In this type, the list items are numbered with upper case roman numbers.
Type "i"	In this type, the list items are numbered with lower case roman numbers.
Type "A"	In this type, the list items are numbered with upper case letters.
Type "a"	In this type, the list items are numbered with lower case letters.

HTML Ordered List Example

Let's see the example of HTML ordered list that displays 4 topics in numbered list. Here we are not defining type="1" because it is the default type.

```
<ol>
    <li>HTML</li>
    <li>Java</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ol>
```

1. HTML
2. Java
3. JavaScript
4. SQL

ol type="I"

Let's see the example to display list in roman number uppercase.

```
<ol type="I">
    <li>HTML</li>
    <li>Java</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ol>
```

Output:

I. HTML
II. Java
III. JavaScript
IV. SQL

ol type="i"

Let's see the example to display list in roman number lowercase.

```
<ol type="i">
    <li>HTML</li>
    <li>Java</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ol>
```

Output:

i. HTML
ii. Java
iii. JavaScript
iv. SQL

ol type="A"

Let's see the example to display list in alphabet uppercase.

```
<ol type="A">
    <li>HTML</li>
    <li>Java</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ol>
```

Output:

A. HTML
B. Java
C. JavaScript
D. SQL

ol type="a"

Let's see the example to display list in alphabet lowercase.

```
<ol type="a">
    <li>HTML</li>
    <li>Java</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ol>
```

Output:

a. HTML
b. Java
c. JavaScript
d. SQL

Start attribute

The start attribute is used with ol tag to specify from where to start the list items.

<ol type="1" start="5"> : It will show numeric values starting with "5".

<ol type="A" start="5"> : It will show capital alphabets starting with "E".

<ol type="a" start="5"> : It will show lower case alphabets starting with "e".

<ol type="I" start="5"> : It will show Roman upper case value starting with "V".

<ol type="i" start="5"> : It will show Roman lower case value starting with "v".

```
<ol type="i" start="5">
    <li>HTML</li>
    <li>Java</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ol>
```

Output:

v. HTML
vi. Java
vii. JavaScript
viii. SQL

reversed Attribute:

This is a Boolean attribute of HTML <ol> tag, and it is new in HTML5 version. If you use the reversed attribute with<ol reversed> tag then it will numbered the list in descending order (7, 6, 5, 4.....1).

Example:

```
<ol reversed>
    <li>HTML</li>
    <li>Java</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ol>
```

Output:

HTML Unordered List | HTML Bulleted List

HTML Unordered List or Bulleted List displays elements in bulleted format . We can use unordered list where we do not need to display items in any particular order. The HTML ul tag is used for the unordered list. There can be 4 types of bulleted list:

- disc
- circle
- square
- none

To represent different ordered lists, there are 4 types of attributes in <ul> tag.

Type	Description
Type "disc"	This is the default style. In this style, the list items are marked with bullets.
Type "circle"	In this style, the list items are marked with circles.
Type "square"	In this style, the list items are marked with squares.
Type "none"	In this style, the list items are not marked .

HTML Unordered List Example

```
<ul>
    <li>HTML</li>
    <li>Java</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ul>
```

Output:

- HTML
- Java
- JavaScript
- SQL

```
ul type="circle"
<ul type="circle">
<li>HTML</li>
<li>Java</li>
<li>JavaScript</li>
<li>SQL</li>
</ul>
```

Output:

- HTML
- Java
- JavaScript
- SQL

```
ul type="square"
<ul type="square">
<li>HTML</li>
<li>Java</li>
<li>JavaScript</li>
<li>SQL</li>
</ul>
```

Output:

- HTML
- Java
- JavaScript
- SQL

```
ul type="none"
<ul type="none">
<li>HTML</li>
<li>Java</li>
<li>JavaScript</li>
<li>SQL</li>
</ul>
```

Output:

- HTML
- Java
- JavaScript
- SQL

```
<ul style="list-style-type: square;">
    <li>HTML</li>
    <li>Java</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ul>
```

Example:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
    <h2>Thetype attribute with CSS property
    </h2>
  <ul style="list-style-type: square;">
    <li>HTML</li>
    <li>Java</li>
        <li>JavaScript</li>
        <li>SQL</li>
    </ul>
 </body>
</html>
```

Output:

HTML Description List | HTML Definition List

HTML Description List or Definition List displays elements in definition form like in dictionary. The <dl>, <dt> and <dd> tags are used to define description list.

The 3 HTML description list tags are given below:

1. <dl> tag defines the description list.
2. <dt> tag defines data term.
3. <dd> tag defines data definition (description).

```
<dl>
  <dt>HTML</dt>
  <dd>is a markup language</dd>
  <dt>Java</dt>
  <dd>is a programming language and platform</
    dd>
 <dt>JavaScript</dt>
 <dd>is a scripting language</dd>
  <dt>SQL</dt>
  <dd>is a query language</dd>
</dl>
```

Output:

HTML

is a markup language

Java

is a programming language and platform

JavaScript

is a scripting language

SQL

is a query language

HTML Nested List

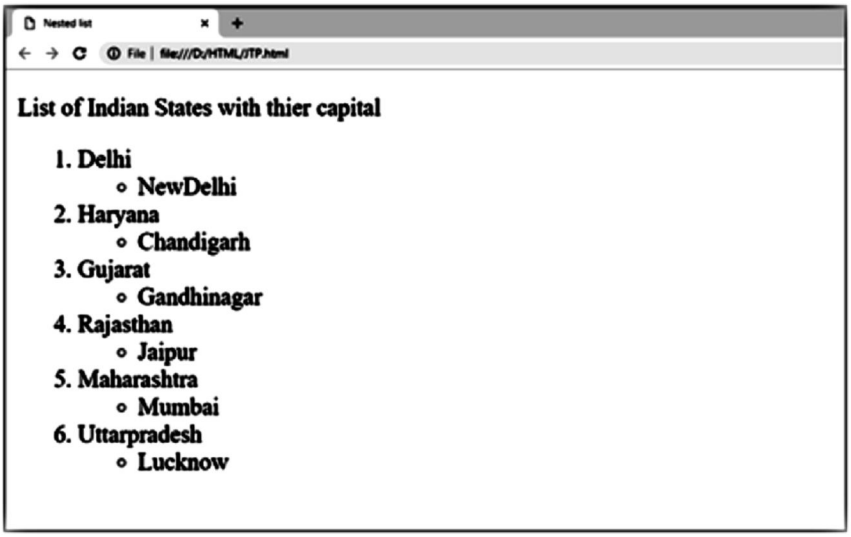
A list within another list is termed as nested list. If you want a bullet list inside a numbered list then such type of list will called as nested list.

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>Nested list</title>
</head>
<body>
    <p>List of Indian States with thier capital</
     p>
<ol>
    <li>Delhi
        <ul>
            <li>NewDelhi</li>
        </ul>
    </li>
    <li>Haryana
        <ul>
            <li>Chandigarh</li>
        </ul>
    </li>
    <li>Gujarat
        <ul>
            <li>Gandhinagar</li>
        </ul>
    </li>
    <li>Rajasthan
        <ul>
            <li>Jaipur</li>
        </ul>
    </li>
    <li>Maharashtra
```

```
        <ul>
            <li>Mumbai</li>
        </ul>
    </li>
    <li>Uttarpradesh
        <ul>
            <li>Lucknow</li></ul>
    </li>
</ol>
</body>
</html>
```

Output:



1.1.8 Tables

Q9. Explain about HTML <table> tag with an example.

Ans : **(Imp.)**

HTML table tag is used to display data in tabular form (row * column). There can be many columns in a row.

We can create a table to display data in tabular form, using <table> element, with the help of <tr>, <td>, and <th> elements.

In Each table, table row is defined by <tr> tag, table header is defined by <th>, and table data is defined by <td> tags.

HTML tables are used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc. But it is recommended to use div tag over table to manage the layout of the page .

HTML Table Tags

Tag	Description
<table>	It defines a table.
<tr>	It defines a row in a table.
<th>	It defines a header cell in a table.
<td>	It defines a cell in a table.
<caption>	It defines the table caption.
<colgroup>	It specifies a group of one or more columns in a table for formatting.
<col>	It is used with <colgroup> element to specify column properties for each column.
<tbody>	It is used to group the body content in a table.
<thead>	It is used to group the header content in a table.
<tfooter>	It is used to group the footer content in a table.

HTML Table Example

Let's see the example of HTML table tag. It output is shown above.

```
<table>
<tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>
<tr><td>Sonoo</td><td>Jaiswal</td><td>60</td></tr>
<tr><td>James</td><td>William</td><td>80</td></tr>
<tr><td>Swati</td><td>Sironi</td><td>82</td></tr>
<tr><td>Chetna</td><td>Singh</td><td>72</td></tr>
</table>
```

Output:

First_Name	Last_Name	Marks
Sonoo	Jaiswal	60
James	William	80
Swati	Sironi	82
Chetna	Singh	72

In the above html table, there are 5 rows and 3 columns = 5 * 3 = 15 values.

HTML Table with Border

There are two ways to specify border for HTML tables.

1. By border attribute of table in HTML
2. By border property in CSS

1. HTML Border attribute

You can use border attribute of table tag in HTML to specify border. But it is not recommended now.

```
<table border="1">
<tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>
<tr><td>Sonoo</td><td>Jaiswal</td><td>60</td></tr>
<tr><td>James</td><td>William</td><td>80</td></tr>
<tr><td>Swati</td><td>Sironi</td><td>82</td></tr>
<tr><td>Chetna</td><td>Singh</td><td>72</td></tr>
</table>
```

Output:

First_Name	Last_Name	Marks
Sonoo	Jaiswal	60
James	William	80
Swati	Sironi	82
Chetna	Singh	72

HTML Table with cell padding

You can specify padding for table header and table data by two ways:

1. By cellpadding attribute of table in HTML
2. By padding property in CSS

The cellpadding attribute of HTML table tag is obselete now. HTML Table width:

We can specify the HTML table width using the **CSS width** property. It can be specify in pixels or percentage.

We can adjust our table width as per our requirement. Following is the example to display table with width.

```
table{
    width: 100%;
}
```

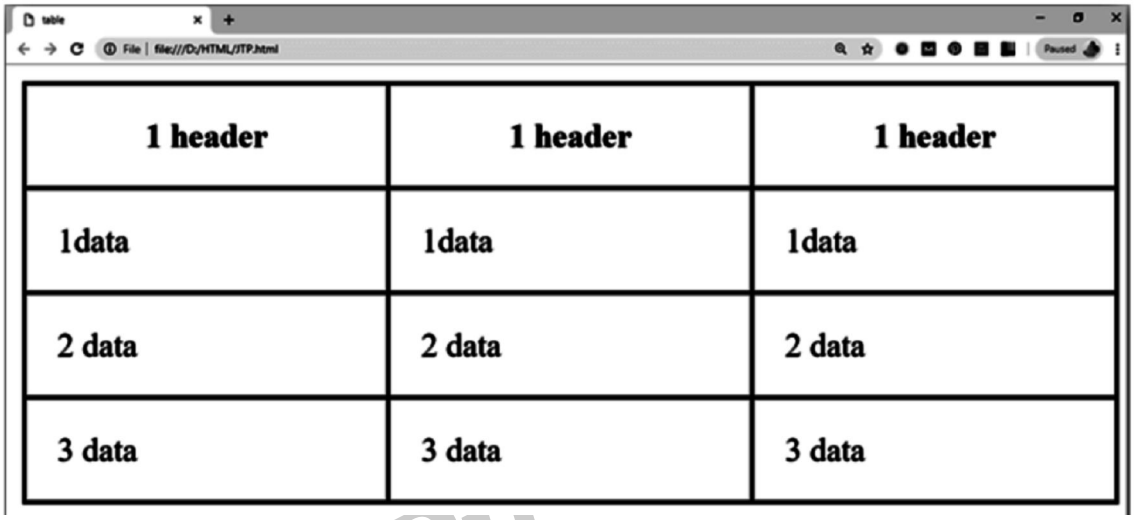
Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>table</title>
    <style>
        table{
            border-collapse: collapse;
            width: 100%;
        }
    th,td{
        border: 2px solid green;
        padding: 15px;
    }

    </style>
  </head>
<body>
  <table>
    <tr>
        <th>1 header</th>
        <th>1 header</th>
        <th>1 header</th>
    </tr>
    <tr>
        <td>1data</td>
        <td>1data</td>
        <td>1data</td>
    </tr>
    <tr>
        <td>2 data</td>
        <td>2 data</td>
        <td>2 data</td>
    </tr>
    <tr>
        <td>3 data</td>
        <td>3 data</td>
```

```
        <td>3 data</td>
    </tr>
</table>
</body>
</html>
```

Output:



1 header	1 header	1 header
1data	1data	1data
2 data	2 data	2 data
3 data	3 data	3 data

HTML Table with colspan

If you want to make a cell span more than one column, you can use the colspan attribute.

It will divide one cell/row into multiple columns, and the number of columns depend on the value of colspan attribute.

Let's see the example that span two columns.

HTML code:

```
<table style="width:100%">
  <tr>
  <th>Name</th>
  <th colspan="2">Mobile No.</th>
  </tr>
  <tr>
  <td>Ajeet Maurya</td>
  <td>7503520801</td>
  <td>9555879135</td>
  </tr>
</table>
```

Output:

Name	Mobile No.	
Ajeet Maurya	7503520801	9555879135

HTML Table with rowspan

If you want to make a cell span more than one row, you can use the rowspan attribute.

It will divide a cell into multiple rows. The number of divided rows will depend on rowspan values.

HTML table with caption

HTML caption is diplayed above the table. It must be used after table tag only.

```
<table>
<caption>Student Records</caption>
<tr><th>First_Name</th><th>Last_Name</th><th>Marks</th></tr>
<tr><td>Vimal</td><td>Jaiswal</td><td>70</td></tr>
<tr><td>Mike</td><td>Warn</td><td>60</td></tr>
<tr><td>Shane</td><td>Warn</td><td>42</td></tr>
<tr><td>Jai</td><td>Malhotra</td><td>62</td></tr>
</table>
```

1.1.9 Frames

Q10. Explain about <frame> tag in HTML.

Ans :

HTML <frame> tag define the particular area within an HTML file where another HTML web page can be displayed.

A <frame> tag is used with <frameset>, and it divides a webpage into multiple sections or frames, and each frame can contain different web pages.

Example 1: Create Vertical frames:

```
<html>
<head>
    <title>Frame tag</title>
</head>
  <frameset cols="25%,50%,25%">
    <frame src="frame1.html" >
    <frame src="frame2.html">
    <frame src="frame3.html">
  </frameset>
</html>
```

Frame1.html

```
<html>
<head>
    <style>
        div{
                                background-color: #7fffd4;
          height: 500px;
        }
    </style>
</head>
<body>
    <div>
        <h2>This is first frame</h2>
    </div>
 </body>
</html>
```

Frame2.html

```
<!DOCTYPE html>
<html>
<head>
    <style>
        div{
```

```
            background-color: #2f4f4f;
            height: 500px;

        }
    </style>
</head>
<body>
    <div>
        <h2>This is Second frame</h2>
    </div>
 </body>
</html>
```

Frame3.html

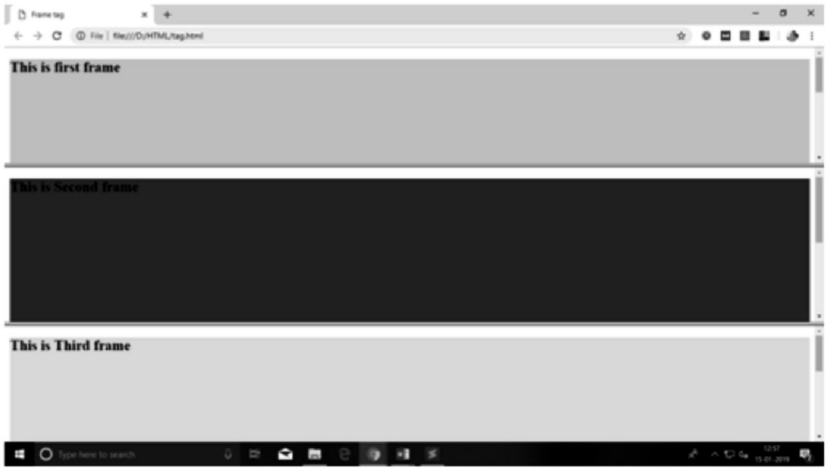
```
<html>
<head>
    <style>
        div{
            background-color: #c1ffc1;
            height: 500px;
                    }
    </style>
</head>
<body>
                <div>
            <h2>This is Third frame</h2>
    </div>
 </body>
</html>
```

Example 2:Create Horizontal frames:

```
<html>
<head>
    <title>Frame tag</title>
</head>
  <frameset rows="30%, 40%, 30%">
```

```
    <frame name="top" src="frame1.html" >
    <frame name="main" src="frame2.html">
    <frame name="bottom" src="frame3.html">
  </frameset>
</html>
```

Output:



Attribute

Tag-specific attribute

Attribute	Value	Description
frameborder	01	It specifies whether to display a border around the frame or not, and its default value is 1
longdsec	URL	It specifies a page which contains the long description of the content of the frame.
marginheight	pixels	It specifies the top and bottom margins of the frame.
marginwidth	pixels	It defines the height of the margin between frames.
name	text	It is used to assign the name to the frame.
noresize	noresize	It is used to prevent resizing of the frame by the user.
scrolling	yesnoauto	It specifies the existence of the scrollbar for overflowing content.
src	URL	It specifies the URL of the document which we want to display in a frame.

1.1.10 Forms

Q11. Explain briefly about <form> tags.

Ans : **(Imp.)**

- An HTML form is a section of a document which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

- An HTML form facilitates the user to enter data that is to be sent to the server for processing such as name, email address, password, phone number, etc. .
- HTML forms are required if you want to collect some data from of the site visitor.

For example: If a user want to purchase some items on internet, he/she must fill the form such as shipping address and credit/debit card details so that item can be sent to the given address.

HTML Form Syntax

```
<form action="server url" method="get|post">
  //input controls e.g. textfield, textarea, radiobutton, button
</form>
```

HTML Form Tags

Tag	Description
<form>	It defines an HTML form to enter inputs by the used side.
<input>	It defines an input control.
<textarea>	It defines a multi-line input control.
<label>	It defines a label for an input element.
<fieldset>	It groups the related element in a form.
<legend>	It defines a caption for a <fieldset> element.
<select>	It defines a drop-down list.
<optgroup>	It defines a group of related options in a drop-down list.
<option>	It defines an option in a drop-down list.
<button>	It defines a clickable button.

HTML <form> element

The HTML <form> element provide a document section to take input from user. It provides various interactive controls for submitting information to web server such as text field, text area, password field, etc.

Syntax:

```
<form>
//Form elements
</form>
```

HTML <input> element

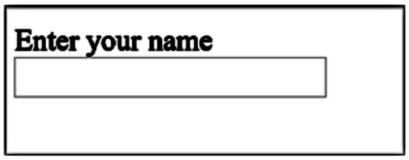
The HTML <input> element is fundamental form element. It is used to create form fields, to take input from user. We can apply different input filed to gather different information form user. Following is the example to show the simple text input.

Example:

```
<body>
  <form>
```

```
    Enter your name  <br>
    <input type="text" name="username">
  </form>
</body>
```

Output:



HTML TextField Control

The type="text" attribute of input tag creates textfield control also known as single line textfield control. The name attribute is optional, but it is required for the server side component such as JSP, ASP, PHP etc.

```
<form>
    First Name: <input type="text" name="firstname"/> <br/>
    Last Name:  <input type="text" name="lastname"/> <br/>
 </form>
```

Output:

HTML <textarea> tag in form

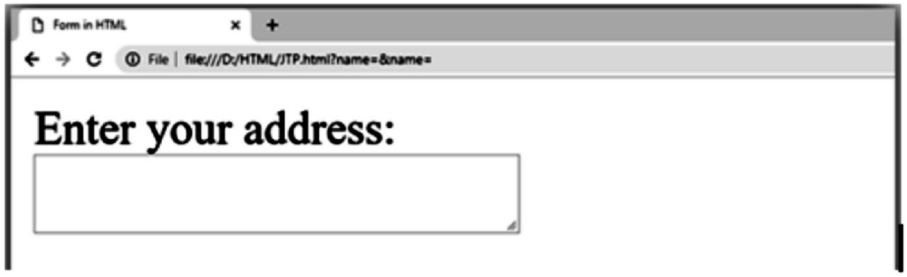
The <textarea> tag in HTML is used to insert multiple-line text in a form. The size of <textarea> can be specify either using "rows" or "cols" attribute or by CSS.

Example:

```
<html>
<head>
    <title>Form in HTML</title>
</head>
<body>
  <form>
```

```
        Enter your address:<br>
      <textarea rows="2" cols="20"></textarea>
  </form>
</body>
</html>
```

Output:



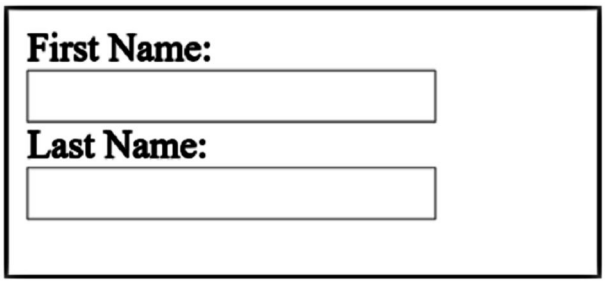
Label Tag in Form

It is considered better to have label in form. As it makes the code parser/browser/user friendly.

If you click on the label tag, it will focus on the text control. To do so, you need to have for attribute in label tag that must be same as id attribute of input tag.

```
<form>
    <label for="firstname">First Name: </label> <br/>
            <input type="text" id="firstname" name="firstname"/> <br/>
  <label for="lastname">Last Name: </label>
            <input type="text" id="lastname" name="lastname"/> <br/>
</form>
```

Output:



HTML Password Field Control

The password is not visible to the user in password field control.

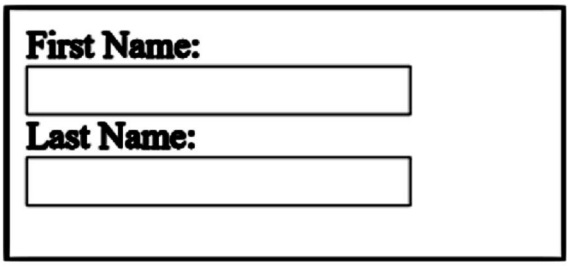
```
<form>
    <label for="password">Password: </label>
```

```
        <input type="password" id="password" name="password"/> <br/>
</form>
```

Output:

```
<form>
    <label for="firstname">First Name: </label> <br/>
        <input type="text" id="firstname" name="firstname"/> <br/>
  <label for="lastname">Last Name: </label>
        <input type="text" id="lastname" name="lastname"/> <br/>
 </form>
```

Output:

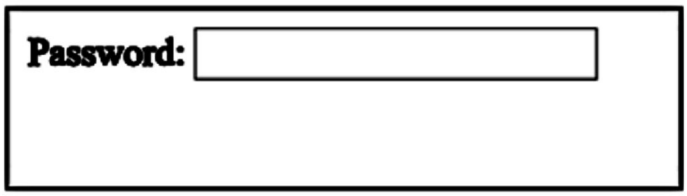


HTML Password Field Control

The password is not visible to the user in password field control.

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

Output:



HTML 5 Email Field Control

The email field in new in HTML 5. It validates the text for correct email address. You must use @ and . in this field.

```
<form>
    <label for="email">Email: </label>
        <input type="email" id="email" name="email"/> <br/>
</form>
```

It will display in browser like below:

Email :

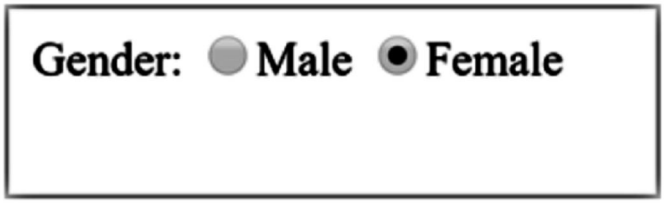
Radio Button Control

The radio button is used to select one option from multiple options. It is used for selection of gender, quiz questions etc.

If you use one name for all the radio buttons, only one radio button can be selected at a time.

Using radio buttons for multiple options, you can only choose a single option at a time.

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

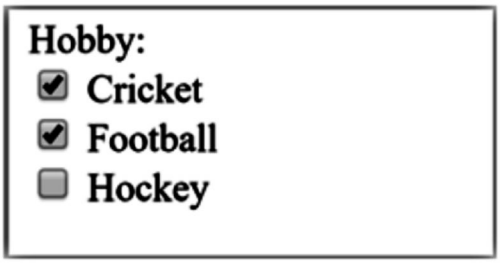


Checkbox Control

The checkbox control is used to check multiple options from given checkboxes.

```
<form>
Hobby:<br>
        <input type="checkbox" id="cricket" name="cricket" value="cricket"/>
            <label for="cricket">Cricket</label> <br>
        <input type="checkbox" id="football" name="football" value="football"/>
            <label for="football">Football</label> <br>
        <input type="checkbox" id="hockey" name="hockey" value="hockey"/>
            <label for="hockey">Hockey</label>
</form>
```

Output:



Submit button control

HTML <input type="submit"> are used to add a submit button on web page. When user clicks on submit button, then form get submit to the server.

Syntax:

<input type="submit" value="submit"**>**

- The type = submit , specifying that it is a submit button
- The value attribute can be anything which we write on button on web page.
- The name attribute can be omit here.

Example:

```
<form>
    <label for="name">Enter name</label><br>
    <input type="text" id="name" name="name"><br>
    <label for="pass">Enter Password</label><br>
    <input type="Password" id="pass" name="pass"><br>
    <input type="submit" value="submit">
</form>
```

Output:

HTML <fieldset> element:

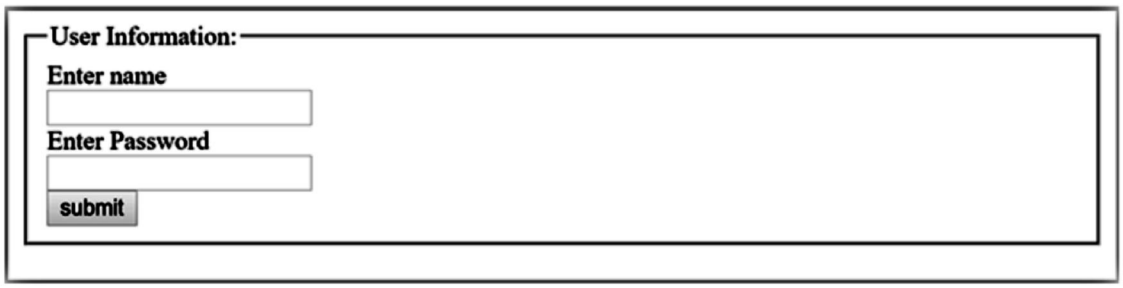
The <fieldset> element in HTML is used to group the related information of a form. This element is used with <legend> element which provide caption for the grouped elements.

Example:

```
<form>
    <fieldset>
      <legend>User Information:</legend>
   <label for="name">Enter name</label><br>
<input type="text" id="name" name="name"><br>
<label for="pass">Enter Password</label><br>
```

<input type="Password" id="pass" name="pass">
<input type="submit" value="submit">
</fieldset>
<./form>

Output:

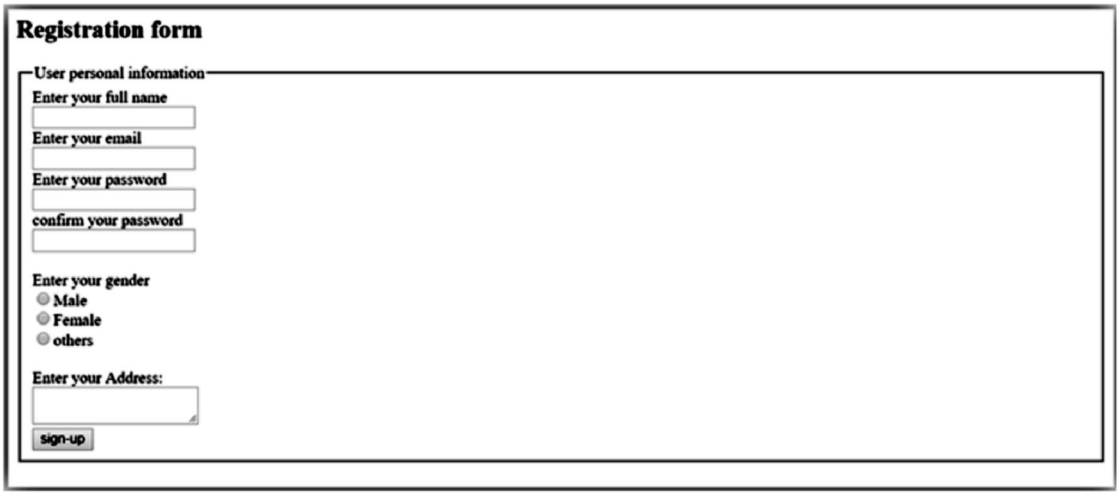


HTML Form Example

```
<html>
<head>
  <title>Form in HTML</title>
</head>
<body>
      <h2>Registration form</h2>
     <form>
      <fieldset>
        <legend>User personal information</legend>
         <label>Enter your full name</label><br>
        <input type="text" name="name"><br>
          <label>Enter your email</label><br>
          <input type="email" name="email"><br>
          <label>Enter your password</label><br>
          <input type="password" name="pass"><br>
          <label>confirm your password</label><br>
          <input type="password" name="pass"><br>
          <br><label>Enter your gender</label><br>
          <input type="radio" id="gender" name="gender" value="male"/>Male <br>
          <input type="radio" id="gender" name="gender" value="female"/>Female <br/>
          <input type="radio" id="gender" name="gender" value="others"/>others <br/>
           <br>Enter your Address:<br>
```

```
            <textarea></textarea><br>
            <input type="submit" value="sign-up">
        </fieldset>
    </form>
  </body>
</html>
```

Output:



Q12. Explain about form input tags.

Ans :

HTML Form Input Types

In HTML <input type=" "> is an important element of HTML form. The "type" attribute of input element can be various types, which defines information field. Such as <input type="text" name="name"> gives a text box.

Following is a list of all types of <input> element of HTML.

type=" "	Description
text	Defines a one-line text input field
password	Defines a one-line password input field
submit	Defines a submit button to submit the form to server
reset	Defines a reset button to reset all values in the form.
radio	Defines a radio button which allows select one option.
checkbox	Defines checkboxes which allow select multiple options form.
button	Defines a simple push button, which can be programmed to perform a task on an event.
file	Defines to select the file from device storage.
image	Defines a graphical submit button.

Following is the description about types of <input> element with examples.

<input type="text">:

<input> element of type "text" are used to define a single-line input text field.

Example:

```
<form>
    <label>Enter first name</label><br>
    <input type="text" name="firstname"><br>
    <label>Enter last name</label><br>
    <input type="text" name="lastname"><br>
    <p><strong>Note:</strong>The default maximum cahracter lenght is 20.</p>
</form>
```

Output:

Input "text" type:

The "text"field defines a sinlge line input text field.

Enter first name

Enter last name

Note: The default maximum cahracter lenght is 20.

<input type="password">:

The <input> element of type "password" allow a user to enter the password securely in a webpage. The entered text in password filed converted into "*" or ".", so that it cannot be read by another user.

Example:

```
<form>
    <label>Enter User name</label><br>
    <input type="text" name="firstname"><br>
    <label>Enter Password</label><br>
    <input type="Password" name="password"><br>
    <br><input type="submit" value="submit">
</form>
```

Output:

Input "password" type:

The "password" field defines a sinlge line input password field to enter the password securely.

Enter User name

Enter Password

submit

<input type="submit">:

The <input> element of type "submit" defines a submit button to submit the form to the server when the "click" event occurs.

Example:

```
<form action="https://www.javatpoint.com/html-tutorial">
    <label>Enter User name</label><br>
    <input type="text" name="firstname"><br>
    <label>Enter Password</label><br>
    <input type="Password" name="password"><br>
    <br><input type="submit" value="submit">
</form>
```

Output:

Input "submit" type:

Enter User name

Enter Password

submit

After clicking on submit button, this will submit the form to server and will redirect the page to action value.We will learn about "action" attribute in later chapters

<input type="reset">:

The <input> type "reset" is also defined as a button but when the user performs a click event, it by default reset the all inputted values.

Example:

```
<form>
    <label>User id: </label>
     <input type="text" name="user-id" value="user">
```

```
        <label>Password: </label>
    <input type="password" name="pass" value="pass"><br><br>
    <input type="submit" value="login">
     <input type="reset" value="Reset">
</form>
```

Output:

Input "reset" type:

Try to change the input values of user id and password, then when you click on reset, it will reset input fields with default values.

<input type="radio">:

The <input> type "radio" defines the radio buttons, which allow choosing an option between a set of related options. At a time only one radio button option can be selected at a time.

Example:

```
<form>
    <p>Kindly Select your favorite color</p>
    <input type="radio" name="color" value="red"> Red <br>
    <input type="radio" name="color" value="blue"> blue <br>
    <input type="radio" name="color" value="green">green <br>
    <input type="radio" name="color" value="pink">pink <br>
    <input type="submit" value="submit">
</form>
```

Output:

Input "radio" type

Kindly Select your favorite color

- ○ Red
- ○ blue
- ○ green
- ○ pink

submit

<input type="checkbox">:

The <input> type "checkbox" are displayed as square boxes which can be checked or unchecked to select the choices from the given options.

Example:

```
<form>
    <label>Enter your Name:</label>
    <input type="text" name="name">
    <p>Kindly Select your favourite sports</p>
    <input type="checkbox" name="sport1" value="cricket">Cricket<br>
    <input type="checkbox" name="sport2" value="tennis">Tennis<br>
    <input type="checkbox" name="sport3" value="football">Football<br>
    <input type="checkbox" name="sport4" value="baseball">Baseball<br>
    <input type="checkbox" name="sport5" value="badminton">Badminton<br><br>
    <input type="submit" value="submit">
</form>
```

Output:

Input "checkbox" type

Registration Form

Enter your Name:

Kindly Select your favorite sports

- ☐ Cricket
- ☐ Tennis
- ☐ Football
- ☐ Baseball
- ☐ Badminton

submit

<input type="button">:

The <input> type "button" defines a simple push button, which can be programmed to control a functionally on any event such as, click event.

Example:

```
1. <form>
2. <input type="button" value="Clcik me" onclick="alert('you are learning HTML')">
3. </form>
```

<input type="file">:

The <input> element with type "file" is used to select one or more files from user device storage. Once you select the file, and after submission, this file can be uploaded to the server with the help of JS code and file API.

Example:

```
<form>
    <label>Select file to upload:</label>
    <input type="file" name="newfile">
    <input type="submit" value="submit">
</form>
```

Output:

Input "file" type.

We can choose any type of file until we do not specify it! The selected file will appear at next to "choose file" option

<input type="image">:

The <input> type "image" is used to represent a submit button in the form of image.

Example:

```
<html>
<body>
<h2>Input "image" type.</h2>
<p>We can create an image as submit button</p>
  <form>
    <label>User id:</label><br>
     <input type="text" name="name"><br><br>
     <input type="image" alt="Submit" src="login.png"  width="100px">
  </form>
   </body>  </html>
```

<input type="url">:

The <input> element of type "url" creates an input filed which enables user to enter the URL.

Example:

```
<form>
    <label>Enter your website URL: </label>
    <input type="url" name="website" placeholder="http://example.com"><br>
    <input type="submit" value="send data">
</form>
```

Output:

Input "url" type

Enter your website URL:

<input type="search">:

The <input> type "search" creates an input filed which allows a user to enter a search string. These are functionally symmetrical to the text input type, but may be styled differently.

Example:

```
<form>
    <label>Search here:</label>
    <input type="search" name="q">
    <input type="submit" value="search">
</form>
```

Output:

Input "search" type

Search here: 

number in given format and it is required to enter the number in input field.

1.1.11 Internal linking

Q13. Explain with an example how to inter link the document in HTML.

Ans : **(Imp.)**

HTML internal link is linked within the same web page. This link can be an absolute path or relative path.

HTML internal link name is followed by the hash sign(#). You have to assign an id to refer section of your page, which is referred to as an internal link to the same page.

When you click on an internal anchor link, you will scroll automatically to the referred section and display it on your browser.

To understand internal link see the below examples.

- Lession.1 link can be referred as Introduction of Lession.1 automatically.
- Lession.2 link can be referred as Introduction of Lession.2 automatically.

Example

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<a href="#lession1">Lession.1</a><br />
<a href="#lession2">Lession.2</a><br />
<a href="#lession3">Lession.3</a><br />
<a href="#lession4">Lession.4</a><br />
<br />
<a id="lession1">Introduction of Lession.1</a>
<p>This is sub topic.1</p>
<p>This is sub topic.2</p>
<p>This is sub topic.3</p>
<p>This is sub topic.4</p>
<br />
<br />
<div id="lession2">Introduction of Lession.2</div>
<p>This is sub topic.1</p>
<p>This is sub topic.2</p>
<p>This is sub topic.3</p>
<p>This is sub topic.4</p>
<br />
<br />
<p id="lession3">Introduction of Lession.3</p>
<p>This is sub topic.1</p>
<p>This is sub topic.2</p>
<p>This is sub topic.3</p>
<p>This is sub topic.4</p>
<br />
<br />
<article id="lession4">Introduction of Lession.4</article>
<p>This is sub topic.1</p>
<p>This is sub topic.2</p>
<p>This is sub topic.3</p>
<p>This is sub topic.4</p>
</body>
</html>
```

Linking to parts of other documents

You can set target to specific sections of the other pages by adding the #id at the end of the href. After adding hash mark is known as a "fragment identifier". This helps a lot for example, you can link to the third section of this tutorial from anywhere else, you have to write

<a href="https://way2tutorial.com/html/html_internal_links.php#section-id">

1.1.12 Meta Elements

Q14. What are called meta elements? Explain

Ans :

HTML <meta> tag is used to represent the metadata about the HTML document. It specifies page description, keywords, copyright, language, author of the documents, etc.

The metadata does not display on the webpage, but it is used by search engines, browsers and other web services which scan the site or webpage to know about the webpage.

With the help of meta tag, you can experiment and preview that how your webpage will render on the browser.

The <meta> tag is placed within the <head> tag, and it can be used more than one times in a document.

Syntax:

<meta charset="utf-8">

Following are some specifications about the HTML <meta> tag

Display	**None**
Start tag/End tag	Empty Tag(Only Start tag)
Usage	Document Structural

Following are some specific syntaxes of meta tag which shows the different uses of meta Tag.

1. **<meta** charset="utf-8"**>**

   It defines the character encoding. The value of charset is "utf-8" which means it will support to display any language.

2. **<meta** name="keywords" content="HTML, CSS, JavaScript, Tutorials"**>**

   It specifies the list of keyword which is used by search engines.

3. **<meta** name="description" content="Free Online tutorials"**>**

   It defines the website description which is useful to provide relevant search performed by search engines.

4. **<meta** name="author" content="thisauthor"**>**

   It specifies the author of the page. It is useful to extract author information by Content management system automatically.

5. **<meta** name="refresh" content="50"**>**

   It specifies to provide instruction to the browser to automatically refresh the content after every 50sec (or any given time).

6. **<meta** http-equiv="refresh" content="5; url=https://www.javatpoint.com/html-tags-list"**>**

   In the above example we have set a URL with content so it will automatically redirect to the given page after the provided time.

7. **<meta** name="viewport" content="width=device-width, initial-scale=1.0"**>**

   It specifies the viewport to control the page dimension and scaling so that our website looks good on all devices. If this tag is present, it indicates that this page is mobile device supported.

Example

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="keywords" content="HTML, CSS, JavaScript, Tutorials">
    <meta name="description" content="Free Online tutorials">
    <meta name="author" content="thisauthor">
    <meta http-equiv="refresh" content="5; url=https://www.javatpoint.com/html-tags-list">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<h2>Example of Meta tag</h2>
<p>This example shows the use of meta tag within an HTML document</p>
</body>
</html>
```


1.2 CASCADING STYLE SHEETS

1.2.1 Introduction

Q15. What is CSS? Explain, how CSS can be used in HTML to change the style of the document.

Ans : **(Imp.)**

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

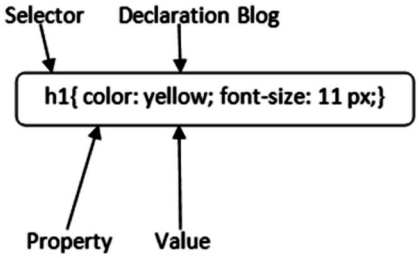
CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

- **CSS saves time:** You can write CSS once and reuse the same sheet in multiple HTML pages.
- **Easy Maintenance:** To make a global change simply change the style, and all elements in all the webpages will be updated automatically.

- **Search Engines:** CSS is considered a clean coding technique, which means search engines won't have to struggle to "read" its content.
- **Superior styles to HTML:** CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Offline Browsing:** CSS can store web applications locally with the help of an offline cache. Using this we can view offline websites.

CSS Syntax

ACSS rule set contains a selector and a declaration block.



Selector: Selector indicates the HTML element you want to style. It could be any tag like <h1>, <title> etc.

Declaration Block: The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

1. color: yellow;
2. font-size: 11 px;

Each declaration contains a property name and value, separated by a colon.

Property: A Property is a type of attribute of HTML element. It could be color, border etc.

Value: Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

Selector{Property1: value1; Property2: value2; .....;}

CSS Selector

CSS selectors are used to select the content you want to style. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

1. CSS Element Selector
2. CSS Id Selector
3. CSS Class Selector
4. CSS Universal Selector
5. CSS Group Selector

1. CSS Element Selector

The element selector selects the HTML element by name.

```
<!DOCTYPE html>
<html>
<head>
<style>
p{
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Output:

This style will be applied on every paragraph.

Me too!

And me!

2. CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

```
<html>
<head>
<style>
#para1 {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<p id="para1">Hello Javatpoint.com</p>
```

```
<p>This paragraph will not be affected.</p>
</body>
</html>
```

Output:

This paragraph will not be affected.

3. CSS Class Selector

The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name.

```
<html>
<head>
<style>
.center {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<h1 class="center">This heading is blue and center-aligned.</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

Output:

CSS Class Selector for specific element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector.

```
<html>
<head>
<style>
p.center {
    text-align: center;
    color: blue;
}
</style>
</head>
```

```
<body>
<h1 class="center">This heading is not affected</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

Output:

This heading is not affected

This paragraph is blue and center-aligned.

4. CSS Universal Selector

The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
    color: green;
    font-size: 20px;
}
</style>
</head>
<body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Output:

This is heading

This style will be applied on every paragraph.

Me too!

And me!

5. CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Let's see the CSS code without group selector.

```
h1 {
    text-align: center;
    color: blue;
}
h2 {
    text-align: center;
    color: blue;
}
p {
    text-align: center;
    color: blue;
}
```

As you can see, you need to define CSS properties for all the elements. It can be grouped in following ways:

```
h1,h2,p {
    text-align: center;
    color: blue;
}
```

Let's see the full example of CSS group selector.

```
<html>
<head>
<style>
h1, h2, p {
    text-align: center;
    color: blue;
}
</style>
</head>
<body>
<h1>Hello Javatpoint.com</h1>
<h2>Hello Java.com (In smaller font)</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

Output:

Hello Java.com

Hello Javatpoint.com (In smaller font)

This is a paragraph.

How to add CSS

CSS is added to HTML pages to format the document according to information in the style sheet. There are three ways to insert CSS in HTML documents.

1. Inline CSS
2. Internal CSS
3. External CSS

1) Inline CSS

Inline CSS is used to apply CSS on a single line or element.

For example:

```
<p style="color:blue">Hello CSS</p>
```

2) Internal CSS

Internal CSS is used to apply CSS on a single document or page. It can affect all the elements of the page. It is written inside the style tag within head section of html.

For example:

```
<style>
p{color:blue}
</style>
```

3) External CSS

External CSS is used to apply CSS on multiple pages or all pages. Here, we write all the CSS code in a css file. Its extension must be .css for example style.css.

For example:

```
p{color:blue}
```

You need to link this style.css file to your html pages like this:

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

The link tag must be used inside head section of html.

1.2.2 Inline styles

Q16. Expalin Inline Style sheets with example.

Ans :

We can apply CSS in a single element by inline CSS technique.

The inline CSS is also a method to insert style sheets in HTML document. This method mitigates some advantages of style sheets so it is advised to use this method sparingly.

If you want to use inline CSS, you should use the style attribute to the relevant tag.

Syntax:

<htmltag style="cssproperty1:value; cssproperty2:value;"> </htmltag>

Example:

<h2 style="color:red;margin-left:40px;">Inline CSS is applied on this heading.</h2>

This paragraph is not affected.

Output:

Inline CSS is applied on this heading.

This paragraph is not affected.

Disadvantages of Inline CSS

- You cannot use quotations within inline CSS. If you use quotations the browser will interpret this as an end of your style value.
- These styles cannot be reused anywhere else.
- These styles are tough to be edited because they are not stored at a single place.
- It is not possible to style pseudo-codes and pseudo-classes with inline CSS.
- Inline CSS does not provide browser cache advantages.

1.2.3 Embedded Style Sheets

Q17. Explain about embedded style sheets

Ans :

Embedded Stylesheet: It allows you to define styles for a particular HTML document as a whole in one place. This is done by embedding the <style></style> tags containing the CSS properties in the head of your document. Embedded style sheets are particularly useful for HTML documents that have unique style requirements from the rest of the documents in your project. However, if the styles need to be applied across multiple documents, you should link to an external style sheet instead of using individual embedded style sheets. Using embedded stylesheets holds a distinct advantage over inline styles which only allow you to address one HTML element at a time.

Syntax: The CSS syntax for embedded style sheets is exactly the same as other CSS code, apart from the fact that it is now wrapped within the <style></style> tags. The <style> tag takes the 'type' attribute that defines the type of style sheet being used (ie. text/CSS).

Example 1:

Below is an HTML document with the CSS styling for the entire web page enclosed within the <style></style> tags. These properties would be applied to all corresponding elements in the HTML document.

```
<html>
<head>
  <title>Page Title</title>
      <!— Embedded stylesheet —>
  <style>
    h2 {
```

```
            font-size: 1.5rem;
            color: #2f8d46;
            text-align: center;
        }

        p {
            font-variant: italic;
        }
    </style>
</head>

<body>
    <h2>Welcome To GFG</h2>
    <p>This document is using an embedded
     stylesheet!</p>
    <p>This is a paragraph</p>
    <p>This is another paragraph</p>
</body>
</html>
```

Output:

Welcome To CFG

Tim document is using an embedded stylesheet!

This is a paragraph

This is another paragraph

All of the <p> elements have been styled using the styling provided in the embedded CSS.

Example 2:

When the list of CSS rule sets is inserted in the style element, it will apply the associated properties to all elements on the web page. In case you want to be more selective and distinctly style an element or a group of elements, use classes and IDs, as shown below.

```
<html>
  <head>
    <title>Page Title</title>
    <!— Embedded stylesheet —>
    <style>
        h2 {
            font-size: 1.5rem;
            color: #2f8d46;
            text-align: center;
        }

        .p-content {
            font-variant: italic;
        }
    </style>
</head>
<body>
    <h2>Welcome To Geeks for Geeks</h2>
    <p class=”p-content”>
        This document is using an
        embedded stylesheet!
    </p>
      <p>This is a paragraph</p>
    <p>This is another paragraph</p>
</body>    </html>
```

Output:

Welcome To GFG

This document is using an embedded stylesheet!

This is a paragraph

This is another paragraph

Specific <p> elements have been styled using the class attribute.

1.2.4 Conflicting Styles

Q18. How CSS will avoid overridden? Explain.

Ans : **(Imp.)**

When more than one set of CSS rules apply to the same element, the browser will have to

decide which specific set will be applied to the element. The rules the browser follows are collectively called Specificity.

Specificity Rules include:

- CSS style applied by referencing external stylesheet has lowest precedence and is overridden by Internal and inline CSS.
- Internal CSS is overridden by inline CSS.
- Inline CSS has highest priority and overrides all other selectors.

Example:

```
<html>
 <head>
    <link rel="stylesheet" type="text/css"
    href="external.css">
    <style type="text/css">
        h1 {
            background-color: red;
            color: white;
        }
        h2 {
            color: blue;
        }
    </style>
</head>
<body>
    <h1>
        Internal CSS overrides external CSS
    </h1>
    <h2 style="color: green;">
        Inline CSS overrides internal CSS
    </h2>
</body>
</html>
```

"external.css" of Example-1:

```
h1{
    background-color: lightgreen;
}
h2{
    color: pink;
}
```

Output:

inline internal and external css

Specificity Hierarchy : Every element selector has a position in the Hierarchy.

1. **Inline style:** Inline style has highest priority.
2. **Identifiers(ID):** ID have the second highest priority.
3. **Classes, pseudo-classes and attributes:** Classes, pseudo-classes and attributes are come next.
4. **Elements and pseudo-elements:** Elements and pseudo-elements have lowest priority.

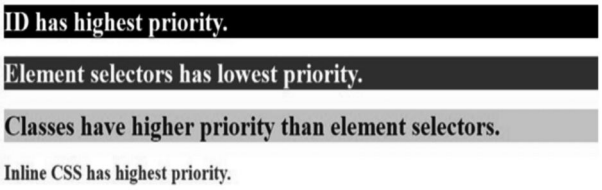
Example-2:

```
<html>
<head>
    <style type="text/css">
        h1 {
            background-color: red;
            color: white;
        }
        #second {
            background-color: black;
            color: white;
        }
        .third {
            background-color: pink;
            color: blue;
        }
```

```
        #second1 {
            color: blue;
        }

        .third1 {
            color: red;
        }
    </style>
</head>
<body>
    <h1 id="second" class="third">
        ID has highest priority.
    </h1>
    <h1>
        Element selectors has lowest priority.
    </h1>
    <h1 class="third">
        Classes have higher priority than element selectors.
    </h1>
    <h2 style="color: green;" id="second1" class="third1">
        Inline CSS has highest priority. </ h2>
</body>
</html>
```

Output:



Specificity Hierarchy

1.2.5 Linking External Sheets

Q19. How can we link external style sheets? Explain.

Ans :

The external style sheet is generally used when you want to make changes on multiple pages. It is ideal for this condition because it facilitates you to change the look of the entire web site by changing just one file.

It uses the <link> tag on every pages and the <link> tag should be put inside the head section.

Example:

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

The external style sheet may be written in any text editor but must be saved with a .css extension. This file should not contain HTML elements.

Let's take an example of a style sheet file named "mystyle.css".

File: mystyle.css

```
body {
    background-color: lightblue;
}
h1 {
    color: navy;
    margin-left: 20px;
}
```

Note: You should not use a space between the property value and the unit. For example: It should be margin-left:20px not margin-left:20 px.

1.2.6 Position Elements

Q20. Out CSS position property.

Ans :

The CSS position property is used to set position for an element. it is also used to place an element behind another and also useful for scripted animation effect.

You can position an element using the top, bottom, left and right properties. These properties can be used only after position property is set first. A position element's computed position property is relative, absolute, fixed or sticky.

Let's have a look at following CSS positioning:

1. CSS Static Positioning
2. CSS Fixed Positioning
3. CSS Relative Positioning
4. CSS Absolute Positioning

1. CSS Static Positioning

This is a by default position for HTML elements. It always positions an element according to the normal flow of the page. It is not affected by the top, bottom, left and right properties.

2. CSS Fixed Positioning

The fixed positioning property helps to put the text fixed on the browser. This fixed test is positioned relative to the browser window, and doesn't move even you scroll the window.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.pos_fixed {
    position: fixed;
    top: 50px;
    right: 5px;
    color: blue;
}
</style>
</head>
<body>
<p>Some text...</p><p>Some text...</p><p>Some text...</p><p>.....</p><p>.... ...</p
><p>.....</p><p>.....</p><p>.....</p><p>.....</p>
<p>..... </p><p>.....</p><p>.....</p><p>.....</p><p>.....</p>
<p>.....</p><p>.....</p><p>Some text...</p><p>Some text...</p><p>Some text...</p>
<p class="pos_fixed">This is the fix positioned text.</p>
</body>
</html>
```

3. CSS Relative Positioning

The relative positioning property is used to set the element relative to its normal position.

```
<html>
<head>
<style>
h2.pos_left {
    position: relative;
    left: -30px;
}
h2.pos_right {
    position: relative;
    left: 30px;
}
</style>
</head>
<body>
<h2>This is a heading with no position</h2>
<h2 class="pos_left">This heading is positioned left according to its normal position</h2>
<h2 class="pos_right">This heading is positioned right according to its normal position</h2>
<p>The style "left:-30px" subtracts 30 pixels from the element's original left position.</p>
<p>The style "left:30px" adds 30 pixels to the element's original left position.</p>
</body>
</html>
```

4. CSS Absolute Positioning

The absolute positioning is used to position an element relative to the first parent element that has a position other than static. If no such element is found, the containing block is HTML.

With the absolute positioning, you can place an element anywhere on a page.

```
<html>
<head>
<style>
h2 {
    position: absolute;
    left: 150px;
    top: 250px;
}
```

```
</style>
</head>
<body>
<h2>This heading has an absolute position</h2>
<p> The heading below is placed 150px from the left and 250px from the top of the page.</
    p>
</body>
</html>
```

All CSS Position Properties

No.	property	description	values
1)	bottom	It is used to set the bottom margin edge for a positioned box.	auto, length, %, inherit
2)	clip	It is used to clip an absolutely positioned element.	shape, auto, inherit
3)	cursor	It is used to specify the type of cursors to be displayed.	url, auto, crosshair, default, pointer, move, e-resize, ne-resize, nw-resize, n-resize, se-resize, sw-resize, s-resize, w-resize, text, wait, help
4)	left	It sets a left margin edge for a positioned box.	auto, length, %, inherit
5)	overflow	This property is used to define what happens if content overflow an element's box.	auto, hidden, scroll, visible, inherit
6)	position	It is used to specify the type of positioning for an element.	absolute, fixed, relative, static, inherit
7)	right	It is used to set a right margin edge for a positioned box.	auto, length, %, inherit
8)	top	It is used to set a top margin edge for a positioned box.	auto, length, %, inherit
9)	z-index	It is used to set stack order of an element.	number, auto, inherit

1.2.7 Box Model And Text Flow

Q21. Explain CSS Box Model.

Ans :

CSS Box Model

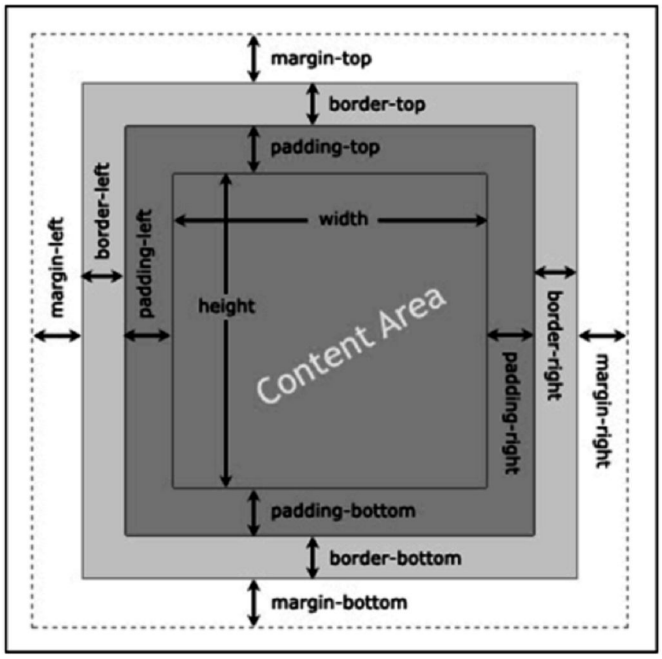
The components that can be depicted on the web page consist of one or more than one rectangular box.

A CSS box model is a compartment that includes numerous assets, such as edge, border, padding and material. It is used to develop the design and structure of a web page. It can be used as a set of tools to personalize the layout of different components. According to the CSS box model, the web browser supplies each element as a square prism.

The following diagram illustrates how the CSS properties of width

- , height
- , padding
- , border
- and margin

dictate that how much space an attribute will occupy on a web page.



The CSSbox model contains the different properties in CSS. These are listed below.

- Border
- Margin
- Padding
- Content

Now, we are going to determine the properties one by one in detail.

Border Field

It is a region between the padding-box and the margin. Its proportions are determined by the width and height of the boundary.

Margin Field

This segment consists of the area between the boundary and the edge of the border.

The proportion of the margin region is equal to the margin-box width and height. It is better to separate the product from its neighbor nodes.

Padding Field

This field requires the padding of the component. In essence, this area is the space around the subject area and inside the border-box. The height and the width of the padding box decide its proportions.

Content Field

Material such as text, photographs, or other digital media is included in this area.

It is constrained by the information edge, and its proportions are dictated by the width and height of the content enclosure.

Elements of the width and height

Typically, when you assign the width and height of an attribute using the CSS width and height assets, it means you just positioned the height and width of the subject areas of that component. The additional height and width of the unit box is based on a range of influences.

The specific area that an element box may occupy on a web page is measured as follows-

Size of the box Properties of CSS

Size of the box	Properties of CSS
Height	height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom
Width	width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

Example 1

```
<!DOCTYPE html>
<head>
<title>CSS Box Model</title>
<style>
        .main
{
            font-size:30px;
            font-weight:bold;
            Text-align:center;
        }
        .gfg
{
            margin-left:50px;
            border:50px solid Purple;
            width:300px;
            height:200px;
            text-align:center;
            padding:50px;
        }
        .gfg1
```

```
{
                font-size:40px;
                font-weight:bold;
                color:black;
                margin-top:60px;
                background-color:purple;
            }
            .gfg2
{
                font-size:20px;
                font-weight:bold;
                background-color:white;
            }
</style>    </head>
<body>
<div class = "main">CSS Box-Model Property</div>
            <div class = "gfg">
            <div class = "gfg1">JavaTpoint</div>
            <div class = "gfg2">A best portal for learn Technologies</div>
</div>
</body>    </html>
```

Output

After the compilation of the above code, you get the following output.

Q22. Explain CSS text-overflow property.

Ans :

CSS text-overflow property

This property specifies the representation of overflowed text, which is not visible to the user. It signals the user about the content that is not visible. This property helps us to decide whether the text should be clipped, show some dots (ellipsis), or display a custom string.

This property does not work on its own. We have to use white-space: nowrap; and overflow: hidden; with this property

Syntax

text-overflow: clip | ellipsis | string | initial | inherit;

Property Values

clip: It is the default value that clips the overflowed text. It truncates the text at the limit of the content area, so that it can truncate the text in the middle of the character.

ellipsis: This value displays an ellipsis (?) or three dots to show the clipped text. It is displayed within the area, decreasing the amount of text.

string: It is used to represent the clipped text to the user using the string of the programmer's choice. It works only in the Firefox browser.

initial: It sets the property to its default value.

inherit: It inherits the property from its parent element.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div{
white-space: nowrap;
height: 30px;
width: 250px;
overflow: hidden;
border: 2px solid black;
font-size: 25px;
}
.jtp{
text-overflow: clip;
}
.jtp1 {
```

```
text-overflow: ellipsis;
}
h2{
color: blue;
}
div:hover {
overflow: visible;
}
p{
font-size: 25px;
font-weight: bold;
color: red;
}
</style>
</head>
<center>
<body>
<p> Hover over the bordered text to see the full content. </p>
<h2>
text-overflow: clip;
</h2>
<div class="jtp">
Welcome to the javaTpoint.com
</div>
<h2>
text-overflow: ellipsis;
</h2>
<div class="jtp1">
Welcome to the javaTpoint.com
</div>
</center>
</body>
</html>
```

Output

1.2.8 Media Types

Q23. Explain CSS media rule property with example.

Ans : **(Imp.)**

One of the most important features of style sheets is that they specify how a document is to be presented on different media: on the screen, on paper, with a speech synthesizer, with a braille device, etc.

We have currently two ways to specify media dependencies for style sheets

- Specify the target medium from a style sheet with the @media or @import at-rules.
- Specify the target medium within the document language.

The @media rule

An @media rule specifies the target media types (separated by commas) of a set of rules.

Given below is an example "

```
<styletyle="text/css">
<!—
    @media print {
      body { font-size: 10pt }
    }
          @media screen {
      body { font-size: 12pt }
    }
    @media screen, print {
      body { line-height: 1.2 }
    }
  —>
      </style>
```

The Document Language

In HTML 4.0, the media attribute on the LINK element specifies the target media of an external style sheet "

Following is an example "

```
<styletyle="text/css">
<!—
<!doctype html public "-//w3c//dtd html 4.0//en">
<html>
<head>
<title>link to a target medium</title>
<link rel = "stylesheet" type = "text/css" media = "print,
    handheld" href = "foo.css">
</head>
<body>
<p>the body...
</body>
</html>
    —>
</style>
```

Recognized Media Types

The names chosen for CSS media types reflect target devices for which the relevant properties make sense. They give a sense of what device the media type is meant to refer to. Given below is a list of various media types "

S.No.	Value & Description
1	**all —**Suitable for all devices.
2	**aural —**Intended for speech synthesizers.
3	**braille—**Intended for braille tactile feedback devices.
4	**embossed—**Intended for paged braille printers.
5	**handheld—**Intended for handheld devices (typically small screen, monochrome, limited bandwidth).
6	**print—**Intended for paged, opaque material and for documents viewed on screen in print preview mode. Please consult the section on paged media.
7	**projection—**Intended for projected presentations, for example projectors or print to transparencies. Please consult the section on paged media.
8	**screen—**Intended primarily for color computer screens.
9	**tty—**Intended for media using a fixed-pitch character grid, such as teletypes, terminals, or portable devices with limited display capabilities.
10	**tv—**Intended for television-type devices.

NOTE " Media type names are case-insensitive.

1.2.9 Building A Css Drop-down Menu

Q24. How to build a drop down menu using CSS. Explain

Ans :

If we want to make a dropdown menu in the Html document using Internal Cascading style sheet, we have to follow the steps which are given below. Using these steps, we can easily make a dropdown menu:

Step 1: Firstly, we have to type the Html code in any text editor or open the existing Html file in the text editor in which we want to use the Internal CSS for making a dropdown menu.

Step 2: Now, we have to place the cursor just after the closing of title tag in the head tag of the Html document and then define the styles inside the <style> tag as shown in the following block.

Now, we have to create the drop-down menu itself. So, first we have to define a class dropbtn with the different attributes which are shown in the following block:

```
<style>
.dropbtn {
    background-color: yellow;
    color: black;
    padding: 10px;
    font-size: 12px;
}
</style>
```

Step 3: Then, we have to use another class which define the dropdown.

```
.dropdown {
    display: inline-block;
    position: relative
}
```

Step 4: Now, we have to create another class which is used to describe how the drop-down menu appears. This class contains various attributes which are shown in the following block.

```
.dropdown-content {
    position: absolute;
    background-color: lightgrey;
    min-width: 200px;
    display: none;
    z-index: 1;
}
```

Step 5: Now we have to add another class for defining the color and size of a text in the drop-down menu.

```
.dropdown-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
}
```

Step 6: And at last in a style tag, we have to edit the hover behaviour of drop-down menu. So, we can easily use the following code in our style tag for editing.

```
.dropdown-content a:hover {
    background-color: orange;
}
.dropdown:hover .dropdown-content {
    display: block;
}
.dropdown:hover .dropbtn {
    background-color: grey;
}
```

Step 7: Now, we have to place the cursor in the body tag where we want to show the drop-down menu. And, then we have to insert the following code in the body tag.

```
<div class="dropdown">
<button class="dropbtn"> JavaTpoint Web Desiging Tutorials</button>
<div class="dropdown-content">
<a href="https://www.javatpoint.com/html-tutorial"> Html </a>
<a href="https://www.javatpoint.com/css-tutorial"> CSS </a>
<a href="https://www.javatpoint.com/javascript-tutorial"> JavaScript </a>
</div>
</div>
```

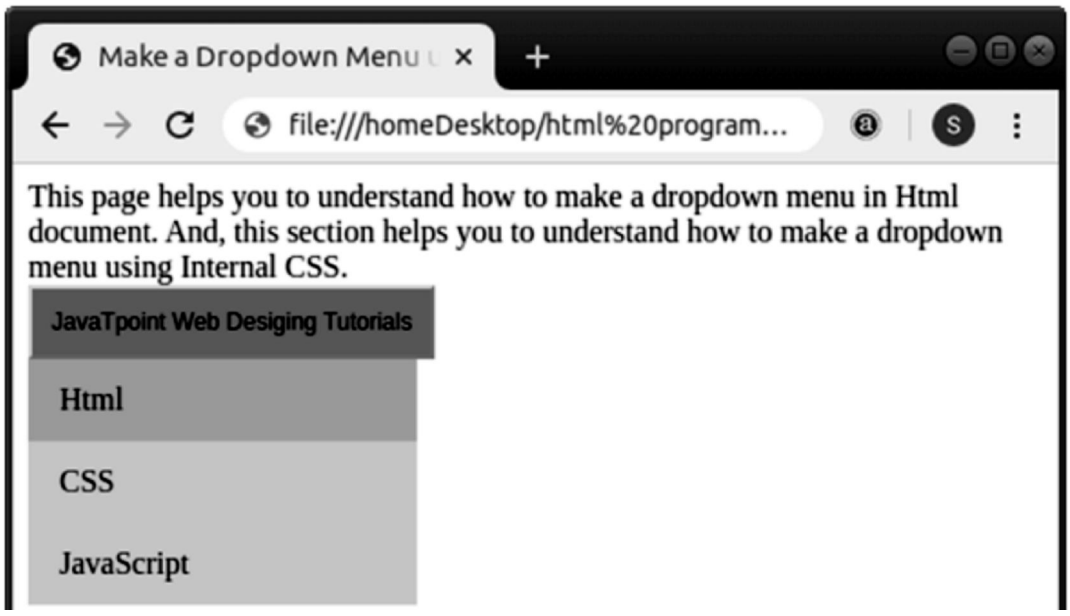
Step 8: And, at last, we have to save the Html file and then run the file in the browser.

```
<!Doctype Html>
<Html>
<Head>
<Title>
Make a Dropdown Menu using Internal CSS
</Title>
```

```
<style>
.dropbtn {
    background-color: yellow;
    color: black;
    padding: 10px;
    font-size: 12px;
}
.dropdown {
    display: inline-block;
    position: relative
}
.dropdown-content {
    position: absolute;
    background-color: lightgrey;
    min-width: 200px;
    display: none;
    z-index: 1;
}
.dropdown-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
}
.dropdown-content a:hover {
background-color: orange;
}
.dropdown:hover .dropdown-content {
display: block;
}
.dropdown:hover .dropbtn {
    background-color: grey;
}
</style>
```

```
</Head>
<Body>
This page helps you to understand how to make a dropdown menu in Html document.
Ans, this section helps you to understand how to make a dropdown menu using Internal CSS.<br>
<div class="dropdown">
<button class="dropbtn"> JavaTpoint Web Desiging Tutorials</button>
<div class="dropdown-content">
<a href="https://www.javatpoint.com/html-tutorial"> Html </a>
<a href="https://www.javatpoint.com/css-tutorial"> CSS </a>
<a href="https://www.javatpoint.com/javascript-tutorial"> JavaScript </a>
</div>
</div>
</Body>
</Html>
```

The output of above Html code is shown in the following screenshot:



1.2.10 User Style Sheets

Q25. Explain CSS User Stylesheets.

Ans :

Some test may require special user style sheets to be applied in order for the case to be verified. In order for proper indications and prerequisite to be displayed every user style sheet should contain the following rules.

```
#user-stylesheet-indication
{
/* Used by the harness to display an indication there is a user
      style sheet applied */
display:block!important;
}
```

The rule #user-stylesheet-indication is to be used by any harness running the test suite.

A harness should identify test that need a user style sheet by looking at their flags meta tag. It then should display appropriate messages indicating if a style sheet is applied or if a style sheet should not be applied.

Harness style sheet rules:

```
.userstyle
{
color:green;
display:none;
}
.nouserstyle
{
color:red;
display:none;
}
```

Harness userstyle flag found:

```
<pid="user-stylesheet-indication"class="userstyle">A user style
sheet is applied.</p>
```

Harness userstyle flag NOT found:

```
<pid="user-stylesheet-indication"class="nouserstyle">A user style
sheet is applied.</p>
```

Within the test case it is recommended that the case itself indicate the necessary user style sheet that is required.

Examples: (code for the cascade.css file)

```
#cascade/* ID name should match user style sheet file name */
{
/* Used by the test to hide the prerequisite */
display:none;
}
```

The rule #cascade in the example above is used by the test page to hide the prerequisite text. The rule name should match the user style sheet CSS file name in order to keep this orderly.

Examples: (code for the cascade-###.xht files)

```
<pid="cascade">
    PREREQUISITE: The <ahref="support/cascade.css">
    "cascade.css"</a> file is enabled as the user agent's user style
    sheet.
</p>
```

The id value should match the user style sheet CSS file name and the user style sheet rule that is used to hide this text when the style sheet is properly applied.

1.2.11 CSS3

Q26. Write about CSS3.

Ans :

CSS3 makes changes to how some visual elements are implemented and rendered by a browser. However, it is not a single hugely unwieldy specification, unlike CSS2. CSS3 is separated into separate modules to facilitate development. This means that the specification comes out in chunks, with more stable modules than others.

Some would be ready for recommendation, while others would be marked as under development drafts, the most recent of which were published as early as June 1999.

Some of the major modules of CSS3 are:

- Box model
- Image values and replaced content
- Text effects
- Selectors
- Backgrounds and borders
- Animations
- User interface (UI)
- Multiple column layout
- 2D/3D transformations

Short Question & Answers

1. What is HTML?

Ans :

HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications.

- **Hyper Text:** HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.
- **Markup language:** A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.
- **Web Page:** A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages.

2. usage of <img> tag.

Ans :

HTML img tag is used to display image on the web page. HTML img tag is an empty tag that contains attributes only, closing tags are not used in HTML image element.

Let's see an example of HTML image.

```
<h2>HTML Image Example</h2>
<img src="good_morning.jpg" alt="Good Morning Friends"/>
```

3. Special characers/ character entieies in HTML.

Ans :

HTML character entities are used as a replacement of reserved characters in HTML. You can also replace characters that are not present on your keyboard by entities.

These characters are replaced because some characters are reserved in HTML. HTML entities provide a wide range of characters which can allow you to add icons, geometric shapes, mathematical operators, etc.

For example, if you use less than (<) or greater than (>) symbols in your text, the browser can mix them with tags that's why character entities are used in HTML to display reserved characters.

4. Write about <hr> tag in HTML.

Ans :

HTML <hr> tag is used to specify a paragraph-level thematic break in HTML document. It is used when you abruptly change your topic in your HTML document. It draw a horizontal line between them. It is also called a Horizontal Rule in HTML.

```
HTML hr tag
<h2>HTML</h2>
<p>HTML is a language for describing web pages.</p>
<hr/>
<h2>HR Tag </h2>
<p> HR tag is used to draw a horizontal line within the texts to sepate content.<p>
```

Output:

HTML

HTML is a language for describing web pages.

HR Tag

HR tag is used to draw a horizontal line within the texts to separate content.

5. Explain about HTML <table> tag with an example.

Ans :

HTML table tag is used to display data in tabular form (row * column). There can be many columns in a row.

We can create a table to display data in tabular form, using <table> element, with the help of <tr>, <td>, and <th> elements.

In Each table, table row is defined by <tr> tag, table header is defined by <th>, and table data is defined by <td> tags.

HTML tables are used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc. But it is recommended to use div tag over table to manage the layout of the page.

6. <form> tags.

Ans :

- An HTML form is a section of a document which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.
- An HTML form facilitates the user to enter data that is to be sent to the server for processing such as name, email address, password, phone number, etc. .
- HTML forms are required if you want to collect some data from of the site visitor.

For example: If a user want to purchase some items on internet, he/she must fill the form such as shipping address and credit/debit card details so that item can be sent to the given address.

HTML Form Syntax

```
<form action="server url" method="get|post">
  //input controls e.g. textfield, textarea, radiobutton, button
</form>
```

HTML Form Tags

Tag	Description
<form>	It defines an HTML form to enter inputs by the used side.
<input>	It defines an input control.
<textarea>	It defines a multi-line input control.
<label>	It defines a label for an input element.
<fieldset>	It groups the related element in a form.
<legend>	It defines a caption for a <fieldset> element.
<select>	It defines a drop-down list.
<optgroup>	It defines a group of related options in a drop-down list.
<option>	It defines an option in a drop-down list.
<button>	It defines a clickable button.

7. Define Internal linking.

Ans :

HTML internal link is linked within the same web page. This link can be an absolute path or relative path.

HTML internal link name is followed by the hash sign(#). You have to assign an id to refer section of your page, which is referred to as an internal link to the same page.

When you click on an internal anchor link, you will scroll automatically to the referred section and display it on your browser.

To understand internal link see the below examples.

- Lession.1 link can be referred as Introduction of Lession.1 automatically.
- Lession.2 link can be referred as Introduction of Lession.2 automatically.

8. Meta Elements.

Ans :

HTML <meta> tag is used to represent the metadata about the HTML document. It specifies page description, keywords, copyright, language, author of the documents, etc.

The metadata does not display on the webpage, but it is used by search engines, browsers and other web services which scan the site or webpage to know about the webpage.

With the help of meta tag, you can experiment and preview that how your webpage will render on the browser.

The <meta> tag is placed within the <head> tag, and it can be used more than one times in a document.

9. What is CSS ?

Ans :

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

- **CSS saves time:** You can write CSS once and reuse the same sheet in multiple HTML pages.
- **Easy Maintenance:** To make a global change simply change the style, and all elements in all the webpages will be updated automatically.

10. Inline Styles.

Ans :

We can apply CSS in a single element by inline CSS technique.

The inline CSS is also a method to insert style sheets in HTML document. This method mitigates some advantages of style sheets so it is advised to use this method sparingly.

If you want to use inline CSS, you should use the style attribute to the relevant tag.

Syntax:

<htmltag style="cssproperty1:value; cssproperty2:value;"> </htmltag>

Example:

```
<h2 style="color:red;margin-left:40px;">Inline CSS is applied on this heading.</h2>
<p>This paragraph is not affected.</p>
```

Output:

Inline CSS is applied on this heading.

This paragraph is not affected.

11. Write about CSS3.

Ans :

CSS3 makes changes to how some visual elements are implemented and rendered by a browser. However, it is not a single hugely unwieldy specification, unlike CSS2. CSS3 is separated into separate modules to facilitate development. This means that the specification comes out in chunks, with more stable modules than others.

Some would be ready for recommendation, while others would be marked as under development drafts, the most recent of which were published as early as June 1999.

Some of the major modules of CSS3 are:

- Box model
- Image values and replaced content
- Text effects
- Selectors
- Backgrounds and borders

- Animations
- User interface (UI)
- Multiple column layout
- 2D/3D transformations

12. CSS User Stylesheets.

Ans :

Some test may require special user style sheets to be applied in order for the case to be verified. In order for proper indications and prerequisite to be displayed every user style sheet should contain the following rules.

```
#user-stylesheet-indication
{
/* Used by the harness to display an indication there is a user
     style sheet applied */
display:block!important;
}
```

The rule #user-stylesheet-indication is to be used by any harness running the test suite.

A harness should identify test that need a user style sheet by looking at their flags meta tag. It then should display appropriate messages indicating if a style sheet is applied or if a style sheet should not be applied.

Harness style sheet rules:

```
.userstyle
{
color:green;
display:none;
}
.nouserstyle
{
color:red;
display:none;
}
```

Harness userstyle flag found:

```
<pid="user-stylesheet-indication"class="userstyle">A user style
sheet is applied.</p>
```

Harness userstyle flag NOT found:

```
<pid="user-stylesheet-indication"class="nouserstyle">A user style
sheet is applied.</p>
```

Within the test case it is recommended that the case itself indicate the necessary user style sheet that is required.

Choose the Correct Answer

1. The correct sequence of HTML tags for starting a webpage is _____. [d]

   (a) Head, Title, HTML, body (b) HTML, Body, Title, Head

   (c0 HTML, Head, Title, Body (d) HTML, Head, Title, Body

2. Which of these elements in HTML can be used for making a text bold? [d]

   (a) <a> (b) <pre>

   (c)
 (d) <b>

3. Which tag do we use in HTML for inserting a line-break? [b]

   (a) <a> (b)

   (c) <b> (d) <pre>

4. Which of these tags helps in the creation of a drop-down box or a combo box? [d]

   (a) <input type = "dropdown"> (b) <list>

   (c) (d) <select>

5. Which HTML tag do we use for displaying the power in the expression, $(x^2 - y^2)$? [c]

   (a) <p> (b) <sub>

   (c) <sup> (d) None of the above

6. In HTML, the correct way of commenting out something would be using: [b]

   (a) ## and # (b) <!– and –>

   (c) </– and -/-> (d) <!– and -!>

7. Which of the following is the correct syntax to make the background-color of all paragraph elements to yellow? [a]

   (a) p {background-color : yellow;} (b) p {background-color : #yellow;}

   (c) all {background-color : yellow;} (d) all p {background-color : #yellow;}

8. Which of the following is the correct syntax to display the hyperlinks without any underline? [c]

   (a) a {text-decoration : underline;} (b) a {decoration : no-underline;}

   (c) a {text-decoration : none;} (d) None of the above

9. Which CSS property and value is used to center an element? [a]

(a) text-align:center (b) align:center

(c) text-align:middle (d) align:middle

10. Which of the following CSS3 property can be used to allow line breaks within words? [d]

(a) line-break (b) line-wrap

(c) word-wrap (d) word-break

Fill in the blanks

1. In HTML, we use the <hr> tag for _____.
2. The non-ASCII characters would be replaced with _____ by the process of URL encoding.
3. HTML tags with no content are called _____.
4. Which _____ attribute specifies where to open the linked document?
5. _____ tag specifies an inline frame?
6. The _____ CSS property used to make the text bold is
7. Inline styles are written within the _____ attribute.
8. The _____ CSS property is used to style the hyperlinks on hover (Mouse over)?
9. The _____ CSS property is used to add shadows to the text?
10. _____ type of CSS is used in the below code?

```
<p style = "border:2px solid red;">
```

ANSWERS

1. horizontal ruler
2. "%"
3. Empty tags
4. target
5. <iframe>
6. font-weight : bold
7. style
8. a:hover
9. text-shadow
10. Inline CSS

UNIT II

Introduction To Java Scripting- introduction, simple program, prompt dialog and alert boxes, memory concepts, operators, decision making, control structures, if… else statement, while, counter-controlled repetitions, switch statement, do… while statement, break and continue statements. Functions – program modules in JavaScript, programmer–defined functions, functions definition, scope rules, global functions, Recursion.

2.1 Introduction To Java Scripting

2.1.1 Introduction

Q1. What is JavaScript? Write about it.

Ans :

Java Script (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

Features of JavaScript

There are following features of Java Script

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

History of Java Script

In 1993, Mosaic, the first popular web browser, came into existence. In the year 1994, Netscape was founded by Marc And reessen. He realized that the web needed to become more dynamic. Thus, a 'glue language' was believed to be provided to HTML to make web designing easy for designers and part-time programmers. Consequently,in1995, the company recruited Brendan Eich intending to implement and embed Scheme programming language to the browser. But, before Brendan could start, the company merged with Sun Micro systems for adding Java into its Navigator so that it could compete with Microsoft over the web technologies and platforms. Now, two languages were there: Java and the scripting language. Further, Netscape decided to give a similar name to the scripting language as Java's. It led to 'Javascript'. Finally, in May 1995, Marc And reessen coined the first code of Javascript named 'Mocha'. Later, the marketing team replaced the name with 'Live Script'. But, due to trademark

reasons and certain other reasons, in December 1995, the language was finally renamed to 'JavaScript'. From then, JavaScript came into existence.

Application of JavaScript

Java Script is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

Java Script Example

```
<script>
document.write("Hello Java Script by
Java Script");
</script>
```

2.1.2 Simple Program

Q2. Explain the simple program structure of Java Script.

Ans : **(Imp.)**

Java script example is easy to code. Java Script provides 3 places to put the Java Script code: within body tag, within head tag and external JavaScript file.

Let's create the first Java Script example.

```
<script type="text/java script">
document.write("JavaScript is a simple
language for javat point learners");
</script>
```

- The script tag specifies that we are using Java Script.
- The text/javascript is the content type that provides information to the browser about the data.

The document.write() function is used to display dynamic content through JavaScript. 3 Places to put JavaScript code

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)

1) Code between the body tag

In the above example, we have displayed the dynamic content using JavaScript. Let's see the simple example of JavaScript that displays alert dialog box.

```
<script type="text/javascript">
alert("Hello Javatpoint");
</script>
```

2) Code between the head tag

Let's see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

In this example, we are creating a function msg(). To create function in JavaScript, you need to write function with function_name as given below.

To call function, you need to work on event. Here we are using onclick event to call msg() function.

```
<html>
<head>
<script type="text/javascript">
function msg(){
alert("Hello Javatpoint");
}
</script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click"
onclick="msg()"/>
</form>
</body>
</html>
```

3. External JavaScript file

We can create external JavaScript file and embed it in many html page.

It provides code re usability because single JavaScript file can be used in several html pages.

An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

Let's create an external Java Script file that prints Hello Javat point in a alert dialog box.

message.js

```
function msg(){
alert("Hello Javatpoint");
}
```

Let's include the JavaScript file into htmlpage. It calls the JavaScript functionon button click.

index.html

```
<html>
<head>
<script type="text/javascript" src="
        message.js"></script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click"
        onclick="msg()"/>
</form>
</body>
</html>
```

2.1.3 Prompt Dialog And Alert Boxes

Q3. How to use prompt() dialog box in Java Script? Explain.

Ans : **(Imp.)**

Java Script prompt() dialog box

The prompt() method in JavaScript is used to display a prompt box that prompts the user for the input. It is generally used to take the input from the user before entering the page. It can be written without using the window prefix. When the prompt box pops up, we have to click "OK" or "Cancel" to proceed.

The box is displayed using the prompt() method, which takes two arguments: The first argument is the label which displays in the text box, and the second argument is the default string, which displays in the textbox. The prompt box consists of two buttons, OK and Cancel. It returns null or the string entered by the user. When the user clicks "OK," the box returns the input value. Otherwise, it returns null on clicking "Cancel".

The prompt box takes the focus and forces the user to read the specified message. So, it should avoid overusing this method because it stops the user from accessing the other parts of the webpage until the box is closed.

Syntax : prompt(message, default)

Values

The parameter values of this function are defined as follows.

Message

It is an optional parameter. It is the text displays to the user. We can omit this value if we don't require to show anything in the prompt.

Default

It is also an optional parameter. It is a string that contains the default value displayed in the textbox.

Example

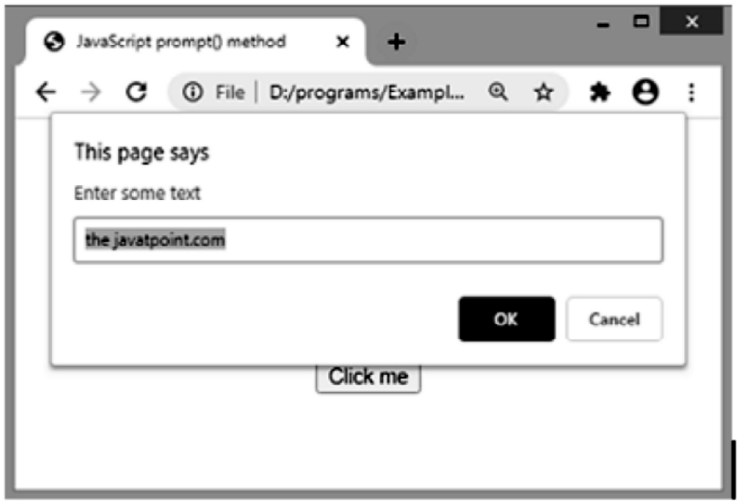
```
<!DOCTYPE html>
<html>      <head>
<title>
JavaScript prompt() method
</title>
<script>
function fun() {
var a = prompt("Enter some text","
     the javatpoint.com");
if (a != null) {
document.getElementById("para"). inner
HTML=" Welcome to" + a;
```

```
}
}
        </script>
        </head>
        <body style = "text-align: center;">
        <h1 style = "color: red;">
        Hello World
        </h1>
        <h2>
      Example of the JavaScript prompt() method
        </h2>
        <button onclick = "fun()">
        Click me
        </button>
        <p id = "para"></p>
      </body>  </html>
```

Output

After the execution of the above code, the output will be -

On clicking the Click me button, the output will be -



After clicking the OK button, the output will be -

Q4. Write about the usage of alert() method in Java Script.

Ans :

Java Script alert ( )

The alert() method in JavaScript is used to display a virtual alert box. It is mostly used to give a warning message to the users. It displays an alert dialog box that consists of some specified message (which is optional) and an OK button. When the dialog box pops up, we have to click "OK" to proceed.

The alert dialog box takes the focus and forces the user to read the specified message. So, we should avoid overusing this method because it stops the user from accessing the other parts of the webpage until the box is closed.

Another example is suppose a user is required to fill the form in which some mandatory fields are required to enter some text, but the user forgets to provide the input. As the part of the validation, we can use the alert dialog box to show a warning message related to fill the textfield.

Rather than showing the warnings or errors, the alert dialog box can be used for normal messages such as 'welcome back', 'Hello XYZ', etc.

Syntax

1. alert(message)

Values

message: It is an optional string that specifies the text to display in the alert box. It consists of the information that we want to show to the users.

Example :

In this example, there is an alert dialog box with a message and an OK button. Here, we are using the line-breaks in the message of the alert box. The line breaks are defined by using the '\n'. The line breaks make the message readable and clear. We have to click the given button to see the effect.

```
<html>
    <head>
  <script type = "text/javascript">
        function fun() {
      alert (" Hello World \n Welcome to the javaTpoint.com \n This is an alert dialog box ");
        }
  </script>
</head>
<body>
    <p> Click the following button to see the effect </p>
    <form>
      <input type = "button" value = "Click me" onclick = "fun();" />
    </form>
</body>
</html>
```

Output

After clicking the button, the output will be -

2.1.4 Memory Concepts

Q5. Explain about the memory allocation concept in Java Script.

Ans : **(Imp.)**

Memory Management! When things (string, array, object) are created, memory is allocated to them in JavaScript. Unlike Low-Level Programming Language C, JavaScript does not have alloc(), malloc(), free() primitive to do a job for them. Memory is automatically freed when that is no longer useful.

This process is called Garbage Collection. Word Automatically brings confusion to developers, that they need not worry about Memory, And this is a mistake, Sir!

In memory management, its life cycle

1. Allocate Memory(explicit or implicit depending on a programming language for us it is implicit)
2. Use it.
3. Release it.

This is a general life cycle.

Allocation Of Memory in JavaScript for memory management

When values are assigned to any variable memory is assigned to it.

```
var a =10;
var b =”CronJ”;
var c ={
      d :1,
      e:’name’
}
var f =[1,2,3,4,5]
var g =newDate();
```

Release when the memory is not needed anymore

High-Level Programming Language like Javascript has some software called Garbage Collector, which does the job of automatic memory deallocation.

Garbage Collection

The main concept of the algorithms designed for garbage collection is the concept of reference. An object can have a reference to another object if the previous object has access to the latter. For example, a JavaScript object can have an implicit reference(when the reference is to its prototypes) and explicit(when the reference is to its properties values).

Below we will explain the algorithms used for Garbage Collection.

1. Reference-counting garbage collection

This algorithm is considered to be the most basic kind of garbage collection algorithm. What these algorithms do is that rather than determining whether any resource is important or not it scans the memory to determine if an object has any other objects referring to it. An object with zero references is considered to be garbage or "collectible".

Example:

```
// Consider the following example
// Declare an object
var object_1 = {
     object_2: {
          object_3: 7
    }
};
     // In this example, create two objects
     // One object is referred by another
     // as one of its properties. Currently,
     // none can be garbage collected
     // The "object_4" variable is the second
     // thing that has a reference to the object
          var object_4 = object_1;
     // The object that was originally in
     // "object_1" has a unique reference
     // embodied by the "object_4" variable
     object_1 = 1;
     //Reference to "object_2" property of
     // the object. This object now has 2
     // references: 1 as a property,
     // The other as the "object_5" variable.
          var object_5 = object_4.object_2;
     // The object that was in "object_1" has
     // now zero references to it. It can be
     // garbage-collected. However its "object_2"
     // property is still referenced by the
     // "object_5" variable, so it cannot be freed.
          object_4 = "Geeks For Geeks";
     // Now the "object_2" property has no
     // references to it and hence it can
     // be garbage collected.
     object_5 = null;
```

1. Obstructions: Circular references

Limitations arise when it comes to circular references. A circular reference occurs when two objects are created with properties that refer each other, thus creating a cycle.

The reference-counting algorithm fails to reclaim the these memory resources as each object has at least one reference pointing to them which prevents both the objects from being marked for garbage collection. Circular references are one of the major cause for memory leaks.

Below example shows an instance of said case.

Example:

```
o
functionDemo() {
varone = {};
     vartwo = {};
// one reference to two
one.object = two;
   // two reference to one
```

```
    two.object = one;
      return'circular';
}
  Demo();
```

2. Mark-and-sweep-algorithm

This algorithm modifies the problem statement from the "object being no longer needed" to the object being "unreachable". This algorithm demands a prerequisite of the knowledge of roots which are a set of objects. In JavaScript, a root is a global object. On a regular basis, the garbage collector starts from these roots and finds all the objects that are referenced from these roots, then all objects referenced from these, etc. Starting from the roots, the garbage collector will find all the objects that are reachable and mark all the non-reachable objects.

Cycles are no longer problem

After the function call returns, the two objects are no longer referenced by any resource that is reachable from the root or global object. Hence, these will be marked as unreachable by the garbage collector and have their allocated memory reclaimed.

Some Limitations

The only limitation that can be found is that it is not possible to explicitly or programmatically trigger garbage collector in JavaScript.

Hence if there are cases when it would be convenient to manually program when to release memory, there are no provisions in JavaScript to trigger such an event.

2.1.5 Operators

Q6. Explain various arithemetic operators used in JavaScript with examples.

Ans : **(Imp.)**

Java Script operators are symbols that are used to perform operations on operands. For example:

```
var sum=10+20;
```

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

1. JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
+ +	Increment	var a=10; a+ +; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

2. Java Script Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
= =	Is equal to	10= =20 = false
= = =	Identical (equal and of same type)	10= =20 = false
!=	Not equal to	10!=20 = true
!= =	Not Identical	20!= =20 = false
>	Greater than	20>10 = true
> =	Greater than or equal to	20> =10 = true
<	Less than	20<10 = false
< =	Less than or equal to	20< =10 = false

3. JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

4. JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

5. JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

6. JavaScript Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator

2.1.6 Decision Making

Q7. Explain various conditional statements used in Java Script.

Ans : **(Imp.)**

Conditional statements are used to perform different actions based on various conditions. The conditional statement evaluates a condition before the execution of instructions.

When you write the code, you require to perform different actions for different decisions. You can easily perform it by using conditional statements.

The conditional statements in JavaScriptare listed below:

- if statement
- if….else statement
- if….else if….statement
- nested if statement
- switch statement

The if statement

It is one of the simplest decision-making statement which is used to decide whether a block of JavaScript code will execute if a certain condition is true.

Syntax

```
if (condition) {
// block of code will execute if the condition is true
}
```

If the condition evaluates to true, the code within if statement will execute, but if the condition evaluates to false, then the code after the end of if statement (after the closing of curly braces) will execute.

For example

```
var x = 78;
if (x>70) {
console.log("x is greater")
}
```

Output

x is greater

The if....else statement

An if….else statement includes two blocks that are if block and else block. It is the next form of the control statement, which allows the execution of JavaScript in a more controlled way. It is used when you require to check two different conditions and execute a different set of codes. The else statement is used for specifying the execution of a block of code if the condition is false.

Syntax

```
if (condition)
{
// block of code will execute if the condition is true
}
 else
 {
// block of code will execute if the condition is false
}
```

If the condition is true, then the statements inside if block will be executed, but if the condition is false, then the statements of the else block will be executed.

For example

```
var x = 40, y=20;
if (x < y)
{
console.log("y is greater");
}
else
{
```

```
    console.log("x is greater");
}
```

The nested if statement

It is an if statement inside an if statement.

Syntax

```
    if (condition1)
{
    Statement 1; //It will execute when condition1 is true
if (condition2)
 {
    Statement 2; //It will execute when condition2 is true
}
else
{
    Statement 3; //It will execute when condition2 is false
}  }
```

Example

```
var num = 20;
if (num > 10)
{
if (num%2==0)
    console.log( num+ " is greater than 10 and even number");
else
    console.log(num+ " is greater than
    10 and odd number");
}
else
{   console.log(num+" is smaller than 10");
}
console.log("After nested if statement");
```

Output

20 is greater than 10 and even number

After nested if statement

The switch statement

It is a multi-way branch statement that is also used for decision-making purposes. In some cases, the switch statement is more convenient than if-else statements. It is mainly used when all branches depend upon the value of a single variable. It executes a block of code depending upon the different cases.

The switch statement uses the break or default keywords, but both of them are optional. Let us define these two keywords:

break: It is used within the switch statement for terminating the sequence of a statement. It is optional to use. If it gets omitted, then the execution will continue on each statement. When it is used, then it will stop the execution within the block.

default: It specifies some code to run when there is no case match. There can be only a single default keyword in a switch. It is also optional, but it is recommended to use it as it takes care of unexpected cases.

If the condition passed to switch doesn't match with any value in cases, then the statement under the default will get executed.

Syntax

```
switch(expression){
case value1:
//code to be executed;
break; //optional
case value2:
//code to be executed;
break;  //optional
.....
default:
 code to be executed if all cases are not matched;
}
```

Example

```
var num = 5;
      switch(num) {
  case 0 : {
    console.log("Sunday");
    break;
  }
  case 1 : {
    console.log("Monday");
    break;
}
  case 2 : {
    console.log("Tuesday");
    break;
}
  case 3 : {
    console.log("Wednesday");
    break;
}
  case 4 : {
    console.log("Thursday");
    break;
}
  case 5 : {
    console.log("Friday");
    break;
}
  case 6 : {
    console.log("Saturday");
    break;
}
  default: {
      console.log("Invalid choice");
      break;
  }
}
```

Output

Friday

2.1.7 Control Structures

Q8. Explain about control structures in Java Script.

Ans : **(Imp.)**

It makes the code compact. It is mostly used in array.

There are four types of loops in Java Script.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

1. Java Script For loop

The Java Script for loop iterates the elements for the fixed number of times. It should be used if number of iteration is known. The syntax of for loop is given below.

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

Let's see the simple example of for loop in javascript.

```
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
```

Output :

1
2
3
4
5

2) JavaScript while loop

The JavaScript while loop iterates the elements for the infinite number of times. It should be used if number of iteration is not known. The syntax of while loop is given below.

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

Let's see the simple example of while loop in javascript.

```
<script>
var i=11;
while (i<=15)
{
document.write(i + "<br/>");
i++;
}
</script>
```

Output:

11
12
13
14
15

3) Java Script do while loop

The JavaScript do while loop iterates the elements for the infinite number of times like while loop. But, code is executed at least once whether condition is true or false. The syntax of do while loop is given below.

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

Let's see the simple example of do while loop in javascript.

```
<script>
var i=21;
do{
document.write(i + "<br/>");
i++;
}while (i<=25);
</script>
```

Output:

21
22
23
24
25

4) JavaScript for in loop

The JavaScript for in loop is used to iterate the properties of an object. We will discuss about it later.

2.1.8 If... Else Statement

Q9. Explain JavaScript If-Else Statement

Ans :

The Java Script if-else statement is used to execute the code whether condition is true or false. There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. if else if statement

Java Script If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

```
if(expression){
//content to be evaluated
}
```

Let's see the simple example of if statement in javascript.

```
<script>
var a=20;
if(a>10){document.write ("value of a is greater than 10");
}
</script>
```

Output of the above example

value of a is greater than 10

Java Script If...else Statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

```
if(expression){
//content to be evaluated if condition is true
}
else{
//content to be evaluated if condition is false
}
```

Let's see the example of if-else statement in JavaScript to find out the even or odd number.

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

Output of the above example

a is even number

Java Script If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

```
if(expression1){
// ontent to be evaluated if expression1 is true
}
else if(expression2){
//content to be evaluated if expression 2 is true
}
else if(expression3){
// content to be evaluated if expression3 is true
}
else{
//content to be evaluated if no expression is true
}
```

Let's see the simple example of if else if statement in javascript.

```
<script>
var a=20;
if(a==10){
document.write("a is equal to 10");
}
else if(a==15){
document.write("a is equal to 15");
}
else if(a==20){
document.write("a is equal to 20");
}
else{
document.write ("a is not equal to 10, 15 or 20");
}
</script>
```

Output of the above example

a is equal to 20

2.1.9 While

Q10. Explain JavaScript While loop with example.

Ans : **(Imp.)**

Java Script while Loop

The syntax of the while loop is:

```
while (condition) {
// body of loop
}
```

Here,

1. A while loop evaluates the condition inside the parenthesis ().
2. If the condition evaluates to true, the code inside the while loop is executed.
3. The condition is evaluated again.
4. This process continues until the condition is false.
5. When the condition evaluates to false, the loop stops.

To learn more about the conditions, visit JavaScript Comparison and Logical Operators.

Example 1: Display Numbers from 1 to 5

```
// program to display numbers from 1 to 5
// initialize the variable
let i = 1, n = 5;
// while loop from i = 1 to 5
while (i <= n) {
console.log(i);
 i += 1;
}
```

Output

```
1
2
3
4
5
```

Example 2: Sum of Positive Numbers Only

```
// program to find the sum of positive numbers
// if the user enters a negative numbers, the loop ends
// the negative number entered is not added to sum
let sum = 0;
// take input from the user
    let number = parseInt(prompt
            ('Enter a number: '));
    while(number >= 0) {
    // add all positive numbers
    sum += number;
// take input again if the number is positive
  number = parseInt(prompt
            ('Enter a number: '));
}
// display the sum
    console.log('The sum is ${sum}.');
```

Output

```
Enter a number: 2
Enter a number: 5
Enter a number: 7
Enter a number: 0
Enter a number: -3
The sum is 14.
```

2.1.10 Counter-controlled Repetitions

11. What are called counter-controlled repetetions? Explain Counter controlled repetition to calculate a class average.

Ans : **(Imp.)**

Counter-controlled repetition is often called definite repetition, because the number of repetitions is known before the loop begins executing.

- A total is a variable in which a script accumulates the sum of a series of values. Variables that store totals should normally be initialized to zero before they are used in a program.
- A counter is a variable a script uses to count.

Uninitialized variables used in mathematical calculations result in logic errors and produce the value NaN (not a number).

- JavaScript represents all numbers as floatingpoint numbers in memory. Floating-point numbers often develop through division. The computer allocates only a fixed amount of space to hold such a value, so the stored floating-point value can only be an approximation.

Pseudocode algorithm that uses counter-controlled repetition to solve the class-average problem.

- Set total to zero
- Set grade counter to one
- While grade counter is less than or equal to ten
- Input the next grade
- Add the grade into the total
- Add one to the grade counter
- Set the class average to the total divided by ten ten
- Print the class average

Counter controlled repetition to calculate a class average

```
<?xml version = "1.0" encoding = "utf-8"?>
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
<html xmlns = http://www.w3.org/1999/xhtml>
<head>
<title> Class Average Program</title>
<script type = "text/javascript" >
<!..
var total;
// sum of grades
var gradeCounter; // number of grades entered
var grade; // grade typed by user (as a string)
var gradeValue; // grade value (converted to integer)
var average; // average of all grades
// Initialization Phase
total = 0; // clear total
gradeCounter = 1; // prepare to loop
// Processing Phase
while ( gradeCounter <= 10 ) // loop 10 times
{
// prompt for input and read grade from user
grade = window.prompt( "Enter integer grade:", "0" );
// convert grade from a string to an integer
gradeValue = parseInt( grade );
// add gradeValue to total
total = total + gradeValue;
// add 1 to gradeCounter
gradeCounter = gradeCounter + 1;
} // end while
// Termination Phase
average = total / 10; // calculate the average
// display average of exam grades
document.writeln(
"Class average is " + average + "" );
// —>
</script>
</head>
<body>
<p> Click Refresh (or Reload) to run the script again<p>
</body></html>
```

2.1.11 Switch Statement

Q12. Explain JavaScript Switch Statement with an example.

Ans : **(Imp.)**

The Java Script switch statement is used in decision making.

The switch statement evaluates an expression and executes the corresponding body that matches the expression's result.

The syntax of the switch statement is:

```
switch(variable/expression) {
  case value1:
    // body of case 1
    break;
  case value2:
    // body of case 2
    break;
  case valueN:
    // body of case N
    break;
  default:
    // body of default
}
```

The switch statement evaluates a variable/ expression inside parentheses ( ).

- If the result of the expression is equal to value1, its body is executed.
- If the result of the expression is equal to value2, its body is executed.
- This process goes on. If there is no matching case, the default body executes.

Example1:

Simple Program Using switch Statement.

```
// program using switch statement
    let a = 2;
switch (a) {
case1:
    a = 'one';
break;
case2:
    a = 'two';
break;
default:
    a = 'not found';
break;
}
console.log('The value is ${a}');
```

Output

The value is two.

In the above program, an expression a = 2 is evaluated with a switch statement.

- The expression's result is evaluated with case 1 which results in false.
- Then the switch statement goes to the second case. Here, the expression's result matches with case 2. So The value is two is displayed.
- The break statement terminates the block and control flow of the program jumps to outside of the switch block.

2.1.12 Do... While Statement

Q13. Explain about JavaScript do...while Loop.

Ans :

The syntax of do...while loop is:

```
do {
// body of loop
} while(condition)
```

Here,

1. The body of the loop is executed at first. Then the condition is evaluated.
2. If the condition evaluates to true, the body of the loop inside the do statement is executed again.
3. The condition is evaluated once again.
4. If the condition evaluates to true, the body of the loop inside the do statement is executed again.
5. This process continues until the condition evaluates to false. Then the loop stops.

Example: Display Numbers from 1 to 5

```
// program to display numbers
    let i = 1;
```

```
const n = 5;
// do...while loop from 1 to 5
    do {
    console.log(i);
    i++;
} while(i <= n);
```

Output

```
1
2
3
4
5
```

2.1.13 Break And Continue Statements

Q14. Explain About Javascript Break And Continue Statements.

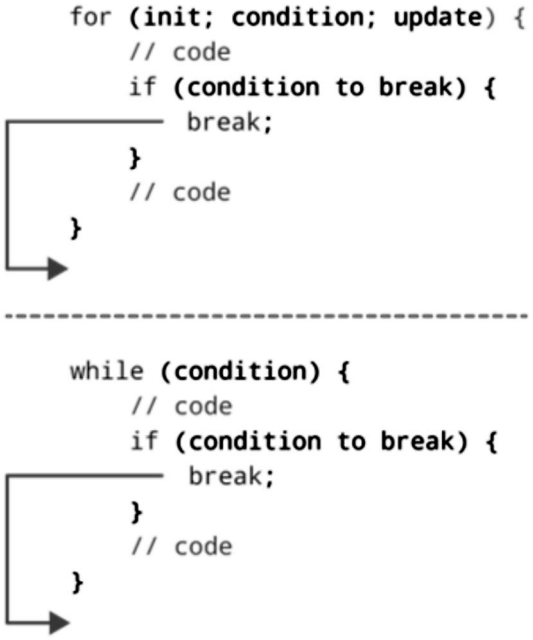
Ans :

The Break Statement Is Used To Terminate The Loop Immediately When It Is Encountered.

The Syntax Of The Break Statement Is:

```
break [label];
```

Working of JavaScript break Statement



Example 1: break with for Loop

```
    // program to print the value of i
    for (let i = 1; i <= 5; i++) {
    // break condition
    if (i == 3) {
    break;
  }
    console.log (i);
}
```

Output

```
1
2
```

In the above program, the for loop is used to print the value of *i* in each iteration. The break statement is used as:

```
if(i == 3) {
break;
}
```

This means, when *i* is equal to 3, the break statement terminates the loop. Hence, the output doesn't include values greater than or equal to 3.

Example 2: break with while Loop

```
    // program to find the sum of positive numbers
    // if the user enters a negative numbers, break ends the loop
    // the negative number entered is not added to sum
        let sum = 0, number;
while(true) {
    // take input again if the number is positive
number = parseInt(prompt('Enter a number: '));
    // break condition
    if(number <0) {
    break;
  }
    // add all positive numbers sum += number;
}
    // display the sum
    console.log('The sum is ${sum}.');
```

Output

```
Enter a number: 1
```

Enter a number: 2

Enter a number: 3

Enter a number: -5

The sum is 6.

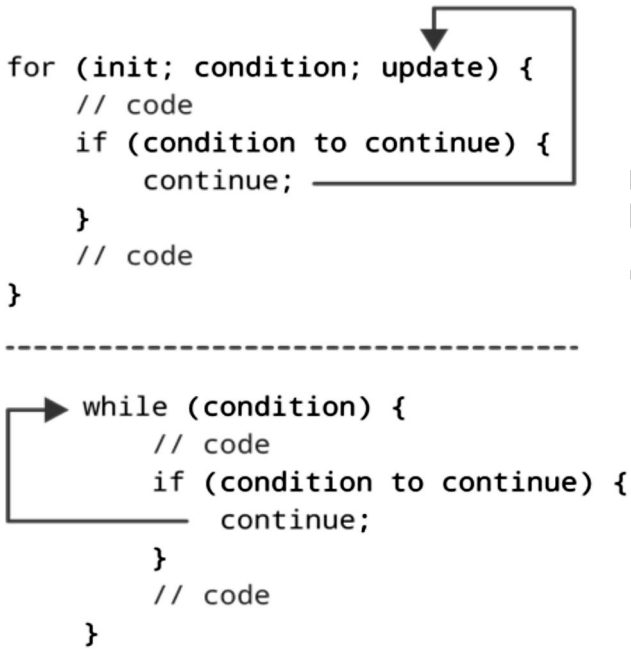
Java Script continue Statement

The continue statement is used to skip the current iteration of the loop and the control flow of the program goes to the next iteration.

The syntax of the continue statement is:

continue [label];

Working of JavaScript continue Statement



In a for loop, continue skips the current iteration and control flow jumps to the update Expression.

Example : Print the Value of i

```
// program to print the value of i
for (let i = 1; i < = 5; i+ +) {
// condition to continue
if (i = = 3) {
continue;
  }
console.log(i);
}
```

Output

1

2

```
4
5
```

In the above program, for loop is used to print the value of i in each iteration.

Notice the continue statement inside the loop.

```
if(i = = 3) {
continue;
}
```

This means

- When i is equal to 3, the continue statement skips the third iteration.
- Then, i becomes 4 andthe test condition and continue statement is evaluated again.
- Hence, 4 and 5 are printed in the next two iterations.

Continue with while Loop

In a while loop, continue skips the current iteration and control flow of the program jumps back to the while condition.

The continue statement works in the same way for while and do...while loops.

Example 2: Calculate Positive Number

```
// program to calculate positive numbers only
// if the user enters a negative number, that number is skipped from calculation
// negative number -> loop terminate
// non-numeric character -> skip iteration
    let sum = 0;
    let number = 0;
    while (number > = 0) {
// add all positive numbers
  sum + = number;
// take input from the user
  number = parseInt(prompt
                    ('Enter a number: '));
// continue condition
    if (isNaN(number)) {
    console.log('You entered a string.');
   number = 0; // the value of number is made 0 again
    continue;
  }
}
// display the sum
    console.log('The sum is ${sum}.');
```

Output

```
Enter a number: 1
Enter a number: 2
Enter a number: hello
You entered a string.
Enter a number: 5
Enter a number: -2
The sum is 8.
```

In the above program, the user enters a number. The while loop is used to print the total sum of positive numbers entered by the user.

2.2 FUNCTIONS

2.2.1 Program Modules In Javascript

Q15. Explain Java script Export and Import Modules

Ans :

We can create modules in JavaScript. In a module, there can be classes, functions, variables, and objects as well. To make all these available in another file, we can use export and import. The export and import are the keywords used for exporting and importing one or more members in a module.

Export

You can export a variable using the export keyword in front of that variable declaration. You can also export a function and a class by doing the same.

Syntax for variable:

```
export let variable_name;
```

Syntax for function:

```
export function function_name() {
 // Statements
}
```

Syntax for class:

```
export class Class_Name {
```

```
    constructor() {
    // Statements
  }
}
```

Example 1

Create a file named export.js and write the below code in that file.

```
    export let num_set = [1, 2, 3, 4, 5];
        export default function hello() {
    console.log("Hello World!");
}
export class Greeting {
    constructor(name) {
        this.greeting = "Hello, " + name;
    }
}
```

Example 2

In this example, we export by specifying the members of the module at the end of the file. We can also use alias while exporting using the as keyword.

```
let num_set = [1, 2, 3, 4, 5];
export default function hello() {
    console.log("Hello World!");
}
class Greeting {
    constructor(name) {
        this.greeting = "Hello, " + name;
    }
}
    export { num_set, Greeting as Greet };
```

Import

You can import a variable using import keyword. You can specify one of all the members that you want to import from a JavaScript file.

Syntax:

import member_to_import from "path_to_js_file";

```
// You can also use an alias while importing a member.
import Greeting as Greet from "./export.js";
// If you want to import all the members but don't
// want to Specify them all then you can do that using
// a ' * ' star symbol.
import * as exp from "./export.js";
```

Example1

Create a file named import.html and write the below code in that file.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Import in ES6</title>
    </head>
    <body>
        <script type="module">
            // Default member first
            import hello, { num_set, Greeting } from "./export.js";
            console.log(num_set);
            hello();
            let g = new Greeting("Aakash");
            console.log(g.greeting);
        </script>
    </body>
</html>
```

2.2.2 Programmer–Defined Functions

Q16. What is function in javascript? Explain about user defined functions.

Ans : **(Imp.)**

A Java Script function is a block of code that consists of a set of instructions to perform a specific task. A function can also be considered as a piece of code that can be used over again and again in the whole program, and it also avoids rewriting the same code. It also helps programmers/coders to divide a huge program into several small functions.

Types of Functions

There are two types of functions in JavaScript like other programming languages such c, c++, and java etc.

- Pre-defined Functions
- User-defined Functions

Here we are going to learn, how to write a user-defined function in JavaScript:

To create a function in JavaScript we have to use the"function " keyword before writing the name of our function as you can see in given syntax:

Syntax of creating function

```
Function functionname( parameters list)
{
  Lines of code to be executed/set of instructions to be executed in order to perform a specific task.
}
```

Before using a function or we can say before calling a function in our program we have to define its definition in between the curly braces. As per your requirement, we can leave the parameter list blank as you can see in the syntax given above.

Example

```
<scripttypescripttype="text/javascript">
<!—
function Hello(){
alert("Hi, there");
}
//—>
</script>
```

How to call the function

We can call the function when we want to use the function in the program by writing its name as you can see below:

```
Hello();
```

Let's see a program in which, we will create a function and use it in the program.

```
<html> <head>
<title>Functions!!!</title>
<script type="text/javascript">
function myfirst Function ( )
{
document.write("This is just a simple user-defined function.<br />");
}
myfirstFunction();
</script>
</head>
<body>
</body> </html>
```

In the above-given program, we have created a function with "myfirstFunction" name and in the definition of this function, we displayed a message "This is just a simple user-defined function" by using the document.write(); function. To print that message, we first need to call the function as you can see in program.

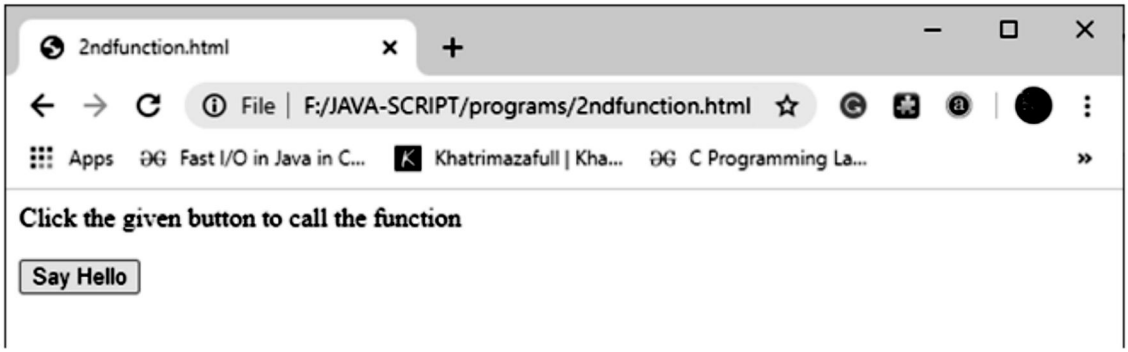
Output

To call the function somewhere else in the script, we just have to write its name as you can see in the given example:

Example

```
<html>  <head>
<script type = "text/javascript">
function sayhi() {
document.write ("Hello there!");
      }
</script> </head>
<body>
<p>Click the given button to call the function</p>
<form>
<input type = "button" onclick = "sayhi()" value = "Say Hello">
</form>
</body></html>
```

Output



Now click on the given button

Function With Parameters

The function we have used in our program is without parameters (or we can say parameter less) because we have not given any parameter in the parameters list and left it empty. But we can also use parameters with function and we can use any number of parameters in our function but they must be separated by comma. These parameters are captured by the function and later any operation can be performed on these parameters inside the function.

Syntax of creating a function with parameters

function function name(parameter1,parameter 2,....parameter N)

{

Lines of code to be executed/set of instructions to be executed in order to perform a specific task.

}

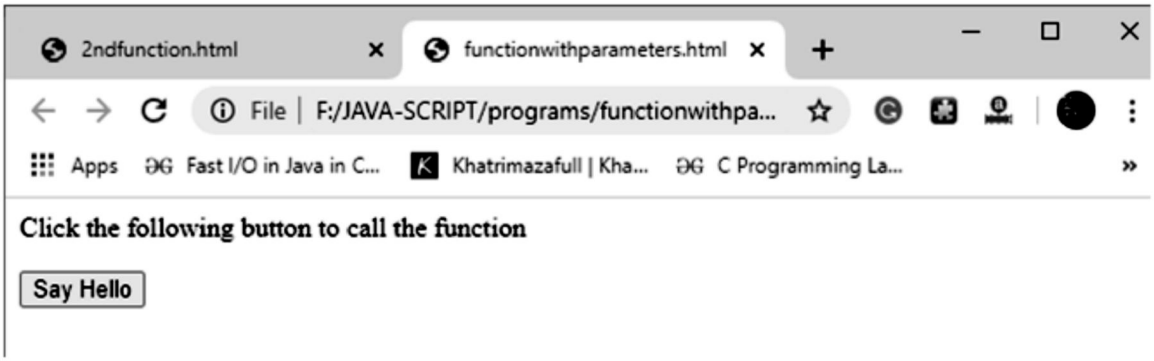
We can understand how to use parameters with function more easily with the help of an example:

Program

```
<html>
<head>
<script type = ”text/javascript”>
function say Hello (name, age,gender) {
document.write (name + ” is ” + age + ” years old” + ” and gender is ” + gender);
      }
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type = ”button” onclick = ”sayHello(‘ Isabella’, 23,’female’)” value = ”Say Hello”>
</form>
</body>  </html>
```

In this program, we created a function named “sayHello ()” with three parameters: name, age, and gender, and defined it in the head section of the HTML document. To use this function, we also created a button using the form tag in the program’s body section and pass the values as arguments. When the user clicks that button, our function is called and gets executed.

Output



Now click on the given button.

Function with return statement

In JavaScript, we can make functions that are able to return a value. To create such type of function, we have to use the return statement, but it must be the last statement in the body of the function (or in the definition of the function). Another essential thing to remember is that we can use only one return statement in a function. If we try to use several return statements in a function, only one return statement is considered, which is reached by the program's control first.

Syntax of function with return statement

```
Function functionname(arg1, arg2)
{
   Set of instructions to be executed
  return val1;

}
```

We can understand how to use the return statement in a function with the help of an example:

Example

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

```
<script type = "text/javascript">
function combine string (string1, string2) { // function1
varcomplete String;
complete String = string1 + string2;
return complete String;
    }
function second Function() {
var result;
result = combine string('Java', 'Script');
document.write (result );
    }
</script>
</head>
<body>
<p>Click the following button to see the function in action</p>
<form>
<input type = "button" onclick = "secondFunction()" value = "Call Function">
</form>
</body>
</html>
```


Explanation of the program

In this program, we have created two functions: combinestring(string1, string2), secondFunction(), and defined their definition in the head section of the HTML document.

Function 1

In the body of "combineString(string1, string2)" function, we created a variable with the name "completestring" to store the string after combining both strings. To return the value stored in this variable, we have used a return statement as you can see in the program.

Function 2

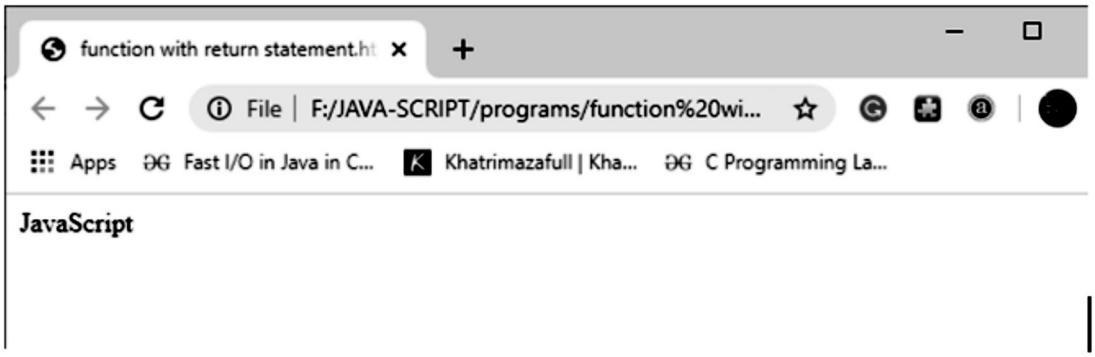
In the body of secondfunction(), we have created a variable that is "result". We have called our first function "completeString(string1, string2)". When the "secondfunction()" is called the "completeString(string1,string2) is also called and the result of this function is stored in the variable "result".

When the execution of the "completeString(string1, string2) function gets completed, the returned value/data gets stored in the "result" variable and in the body of "secondfucntion()" function, we have displayed the value stored in the variable "result" by using document.write() statement.

To call the "secondfunction(),"" we have created a button for the user using the form tag. When the user clicks on that button, our secondfucntion() will be triggered.

Output

Click on the given button.



2.2.3 Functions Definition

Q17. Explain how to define and use a function in Java script.

Ans : **(Imp.)**

A function is a block of code that performs a specific task.

Suppose you need to create a program to create a circle and color it. You can create two functions to solve this problem:

- a function to draw the circle
- a function to color the circle

Dividing a complex problem into smaller chunks makes your program easy to understand and reusable.

JavaScript also has a huge number of inbuilt functions. For example, Math.sqrt() is a function to calculate the square root of a number.

In this tutorial, you will learn about user-defined functions.

Declaring a Function

The syntax to declare a function is:

function nameOfFunction () {

```
    // function body
}
```

- A function is declared using the function keyword.
- The basic rules of naming a function are similar to naming a variable. It is better to write a descriptive name for your function. For example, if a function is used to add two numbers, you could name the function add or addNumbers.
- The body of function is written within {}.

For example,

```
// declaring a function named greet()
functiongreet() {
console.log("Hello there");
}
```

Calling a Function

In the above program, we have declared a function named greet(). To use that function, we need to call it.

Here's how you can call the above greet() function.

```
// function call
greet();
```

Example 1: Display a Text

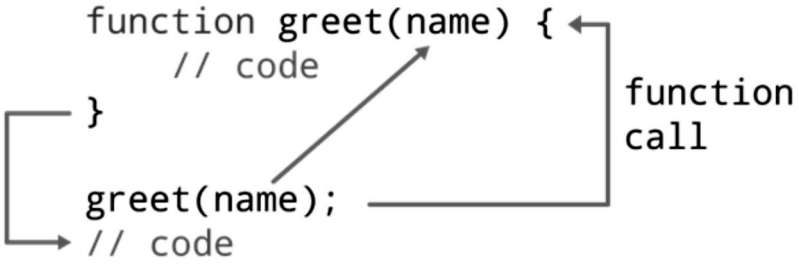
```
// program to print a text
// declaring a function
functiongreet() {
console.log("Hello there!");
}
// calling the function
greet();
```

Output

```
Hello there!
```

Function Parameters

A function can also be declared with parameters. A parameter is a value that is passed when declaring a function.



Working of JavaScript Function with parameter

Example 2: Function with Parameters

```
// program to print the text
// declaring a function
functiongreet(name) {
console.log("Hello " + name + ":)");
}
// variable name can be different
    let name = prompt("Enter a name: ");
// calling function
    greet(name);
```

Output

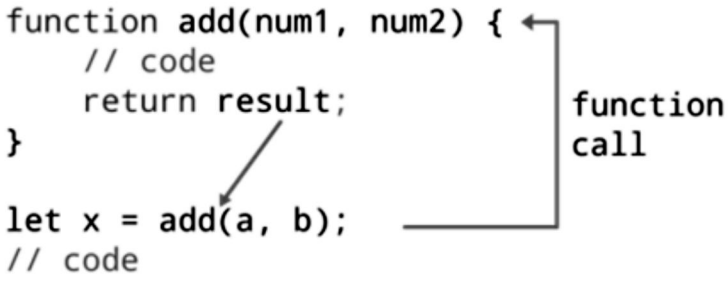
```
Enter a name: Simon
Hello Simon :)
```

In the above program, the greet function is declared with a name parameter. The user is prompted to enter a name. Then when the function is called, an argument is passed into the function.

Function Return

The return statement can be used to return the value to a function call.

The return statement denotes that the function has ended. Any code after return is not executed. If nothing is returned, the function returns an undefined value.



Working of JavaScript Function with return statement

2.2.4 Scope Rules

Q18. Write about the scope rules in javascript

Ans :

A scope can be defined as the region of the execution, a region where the expressions and values can be referenced.

There are two scopes in JavaScript that are global and local:

Global Scope: In the global scope, the variable can be accessed from any part of the JavaScript code.

Local Scope : In the local scope, the variable can be accessed within a function where it is declared.

In the function's body, the precedence of the local variable is more than the global variable with the same name. If the name of the function's local variable is same as the name of the global variable, then the local variable hides the global variable.

Example1:

In this example, we are declaring two variables one variable has a global scope, and the second variable has local scope. Both of the variables are declared with the same name.

In the output, we can see that the variable with locally scoped overrides the value of the global variable.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<script>
var $var12 = 200;
function example() {
var $var12 = 300;
document.write("Inside example() function = " + $var12);
}
document.write("Outside example() function = " + $var12);
document.write("<br>");
example();
</script>
</body>
</html>
```

Output

When we declare a variable inside a function without using the var keyword, it acts as a global variable. Let's see an illustration of the same.

2.2.5 Global Functions

Q19. Explain the usage of global functions in javascript

Ans :

A JavaScript global variable is declared outside the function or declared with window object. It can be accessed from any function.

Let's see the simple example of global variable in JavaScript.

```
<script>
var value=50;//global variable
function a(){
alert(value);
}
function b(){
alert(value);
}
</script>
```

Declaring JavaScript global variable within function

To declare JavaScript global variables inside function, you need to use window object. For example:

```
window.value=90;
```

Now it can be declared inside any function and can be accessed from any function. For example:

```
function m(){
window.value=100;//declaring global variable by window object
}
function n(){
alert(window.value);//accessing global variable from other function
}
```

Internals of global variable in JavaScript

When you declare a variable outside the function, it is added in the window object internally. You can access it through window object also. For example:

```
var value=50;
function a(){
alert(window.value);//accessing global variable
}
```

2.2.6 Recursion

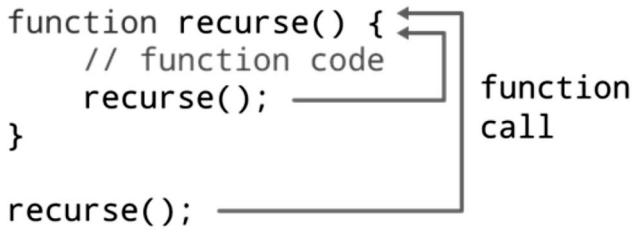
Q20. Explain recursive functions with example

Ans :

Recursion is a process of calling itself. A function that calls itself is called a recursive function. The syntax for recursive function is:

```
functionrecurse() {
// function code
recurse();
// function code
}
recurse();
```

Here, the recurse() function is a recursive function. It is calling itself inside the function.



Working of recursion in JavaScript

A recursive function must have a condition to stop calling itself. Otherwise, the function is called indefinitely.

Once the condition is met, the function stops calling itself. This is called a base condition.

To prevent infinite recursion, you can use if...else statement (or similar approach) where one branch makes the recursive call, and the other doesn't.

So, it generally looks like this.

```
functionrecurse() {
if(condition) {
recurse();
  }
else {
// stop calling recurse()
  }
}
recurse();
```

A simple example of a recursive function would be to count down the value to 1.

Example 1: Print Numbers

```
// program to count down numbers to 1
   functioncountDown(number) {
// display the number
   console.log(number);
// decrease the number value
     const new Number = number - 1;
// base case
if (newNumber >0) {
countDown(newNumber);
  }
}
countDown(4);
```

Output

```
4
3
2
1
```

Short Question & Answers

1. JavaScript.

Ans :

Java Script (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

2. Memory Concept.

Ans :

Memory Management! When things (string, array, object) are created, memory is allocated to them in JavaScript. Unlike Low-Level Programming Language C, JavaScript does not have alloc(), malloc(), free() primitive to do a job for them. Memory is automatically freed when that is no longer useful.

This process is called Garbage Collection. Word Automatically brings confusion to developers, that they need not worry about Memory, And this is a mistake, Sir!

3. Java Script For loop

Ans :

The Java Script for loop iterates the elements for the fixed number of times. It should be used if number of iteration is known. The syntax of for loop is given below.

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

Let's see the simple example of for loop in javascript.

```
    <script>
    for (i=1; i<=5; i++)
{
    document.write(i + "<br/>")
}
    </script>
```

Output :

```
1
2
3
4
5
```

4. Switch Statement.

Ans :

The Java Script switch statement is used in decision making.

The switch statement evaluates an expression and executes the corresponding body that matches the expression's result.

The syntax of the switch statement is:

```
switch(variable/expression) {
  case value1:
    // body of case 1
    break;
  case value2:
    // body of case 2
    break;
  case valueN:
    // body of case N
    break;
  default:
    // body of default
}
```

The switch statement evaluates a variable/expression inside parentheses ( ).

- If the result of the expression is equal to value1, its body is executed.
- If the result of the expression is equal to value2, its body is executed.
- This process goes on. If there is no matching case, the default body executes.

5. What is function in javascript?

Ans :

A Java Script function is a block of code that consists of a set of instructions to perform a specific task. A function can also be considered as a piece of code that can be used over again and again in the whole program, and it also avoids rewriting the same code. It also helps programmers/coders to divide a huge program into several small functions.

Types of Functions

There are two types of functions in JavaScript like other programming languages such c, c++, and java etc.

- Pre-defined Functions
- User-defined Functions

6. scope rules in javascript

Ans :

A scope can be defined as the region of the execution, a region where the expressions and values can be referenced.

There are two scopes in JavaScript that are global and local:

Global Scope: In the global scope, the variable can be accessed from any part of the JavaScript code.

Local Scope : In the local scope, the variable can be accessed within a function where it is declared.

In the function's body, the precedence of the local variable is more than the global variable with the same name. If the name of the function's local variable is same as the name of the global variable, then the local variable hides the global variable.

7. Global functions in javascript.

Ans :

A JavaScript global variable is declared outside the function or declared with window object. It can be accessed from any function.

Let's see the simple example of global variable in JavaScript.

```
<script>
var value=50;//global variable
function a(){
alert(value);
}
function b(){
alert(value);
}
</script>
```

8. Recursion.

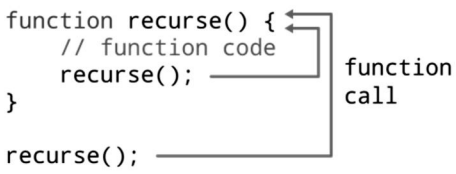
Ans :

Recursion is a process of calling itself. A function that calls itself is called a recursive function.

The syntax for recursive function is:

```
function recurse() {
// function code
    recurse();
// function code
}
recurse();
```

Here, the recurse() function is a recursive function. It is calling itself inside the function.



9. Features of Java Script.

Ans :

1. All popular web browsers support JavaScript as they provide built-in execution environ- ments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

10. Counter-controlled Repetitions.

Ans :

Counter-controlled repetition is often called definite repetition, because the number of repetitions is known before the loop begins executing.

- A total is a variable in which a script accumulates the sum of a series of values. Variables that store totals should normally be initialized to zero before they are used in a program.
- A counter is a variable a script uses to count.

  Uninitialized variables used in mathematical calculations result in logic errors and produce the value NaN (not a number).
- JavaScript represents all numbers as floatingpoint numbers in memory. Floating-point numbers often develop through division. The computer allocates only a fixed amount of space to hold such a value, so the stored floating-point value can only be an approximation.

  Pseudocode algorithm that uses counter-controlled repetition to solve the class-average problem.
- Set total to zero
- Set grade counter to one
- While grade counter is less than or equal to ten
- Input the next grade
- Add the grade into the total
- Add one to the grade counter
- Set the class average to the total divided by ten ten
- Print the class average

11. Explain about JavaScript do...while Loop.

Ans :

The syntax of do...while loop is:

```
do {
// body of loop
} while(condition)
```

Here,

1. The body of the loop is executed at first. Then the condition is evaluated.
2. If the condition evaluates to true, the body of the loop inside the do statement is executed again.
3. The condition is evaluated once again.
4. If the condition evaluates to true, the body of the loop inside the do statement is executed again.
5. This process continues until the condition evaluates to false. Then the loop stops.

Example: Display Numbers from 1 to 5

```
// program to display numbers
    let i = 1;
const n = 5;
// do...while loop from 1 to 5
    do {
    console.log(i);
    i++;
} while(i <= n);
```

Output

```
1
2
3
4
5
```


12. Declaring a Function in Java script.

Ans ;

The syntax to declare a function is:

```
function nameOfFunction () {
// function body
}
```

- A function is declared using the function keyword.
- The basic rules of naming a function are similar to naming a variable. It is better to write a descriptive name for your function. For example, if a function is used to add two numbers, you could name the function add or addNumbers.
- The body of function is written within {}.

13. Explain the simple program structure of Java Script.

Ans :

Java script example is easy to code. Java Script provides 3 places to put the Java Script code: within body tag, within head tag and external JavaScript file.

Let's create the first Java Script example.

```
<script type="text/java script">
document.write("JavaScript is a simple
language for javat point learners");
</script>
```

- The script tag specifies that we are using Java Script.
- The text/javascript is the content type that provides information to the browser about the data.

The document.write() function is used to display dynamic content through JavaScript. 3 Places to put JavaScript code

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)

1) Code between the body tag

In the above example, we have displayed the dynamic content using JavaScript. Let's see the simple example of JavaScript that displays alert dialog box.

```
<script type="text/javascript">
alert("Hello Javatpoint");
</script>
```

2) Code between the head tag

Let's see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

In this example, we are creating a function msg(). To create function in JavaScript, you need to write function with function_name as given below.

To call function, you need to work on event. Here we are using onclick event to call msg() function.

```
<html>
<head>
<script type="text/javascript">
function msg(){
alert("Hello Javatpoint");
}
</script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click"
onclick="msg()"/>
</form>
</body>
</html>
```


3. External JavaScript file

We can create external JavaScript file and embed it in many html page.

It provides code re usability because single JavaScript file can be used in several html pages.

An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

Let's create an external Java Script file that prints Hello Javat point in a alert dialog box.

message.js

```
function msg(){
alert("Hello Javatpoint");
}
```

Let's include the JavaScript file into htmlpage. It calls the JavaScript functionon button click.

index.html

```
<html>
<head>
<script type="text/javascript" src="
        message.js"></script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click"
        onclick="msg()"/>
</form>
</body>
</html>
```

Choose the Correct Answer

1. Which HTML element is used to put the JavaScript code? [d]
   (a) <javascript> (b) <js>
   (c) <scripting> (d) <script>
2. Which of the following methods can be used to display data in some form using Javascript? [d]
   (a) document.write() (b) console.log()
   (c) window.alert() (d) All of the above
3. What keyword is used to check whether a given property is valid or not? [a]
   (a) in (b) is in
   (c) exists (d) lies
4. What will be the output of the following code snippet? [a]
   var a = Math.max();
   var b = Math.min();
   print(a); print(b);
   (a) -Infinity Infinity (b) Infinity -Infinity
   (c) Infinity Infinity (d) -Infinity -Infinity
5. What will be the output of the following code snippet? [b]
   print(typeof(NaN));
   (a) Object (b) Number
   (c) String (d) None of the above
6. What will be the output of the following code snippet? [a]
   print(parseInt("123Hello"));
   print(parseInt("Hello123"));
   (a) 123 NaN (b) 123Hello Hello123
   (c) NaNNaN (d) 123 123
7. In the JavaScript, which one of the following is not considered as an error: [c]
   (a) Syntax error (b) Missing of semicolons
   (c) Division by zero (d) Missing of Bracket
8. Which of the following is the correct syntax to print a page using JavaScript? [ ]
   (a) print(); (b) print();
   (c) print(); (d) print();

9. What happens if the return statement has no related expression? [a]

(a) It will return a undefined value (b) It will throw a exception

(c) It will return the 0 as the value (d It will throw a error

10. When the switch statement matches the expression with the given labels, how is the comparison done? [a]

(a) Both the datatype and the result of the expression are compared.

(b) Only the datatype of the expression is compared.

(c) Only the value of the expression is compared.

(d) None of the above.

Fill in the blanks

1. Javascript is an _____ language?
2. When an operator's value is NULL, the typeof returned by the unary operator is _____.
3. The "function" and " var" are known as _____.
4. When there is an indefinite or an infinite value during an arithmetic computation in a program, then JavaScript prints _____.
5. The execution of a function stops when the program control encounters the _____ statement in the body of the function.
6. If a function which does not return a value is known as _____.
7. mThe process in which an object or data structure is translated into a format suitable for transferral over a network, or storage is called _____.
8. In the following given line of code, the prototype representing the _____.

   functionx(){};
9. What does … operator do in JS?
10. _____ and _____ are the two basic groups of datatypes in JavaScript?

ANSWERS

1. Object-Oriented
2. Object
3. Declaration statements
4. Displays "Infinity"
5. return statement
6. Procedures
7. Object Serialization
8. Prototype of a function
9. It is used to spread iterables to individual elements
10. Primitive andReference types.

UNIT III

Arrays - Introduction, Declaring and Allocating Arrays, References and Reference Parameters, Passing Arrays to Functions. Multidimensional Arrays, **EVENTS –** Registering Event Handling, Event Onload, Onmouseover, Onmouseout, Onfocus, Onblur, Onsubmit, Onreset, Event Bubbling, More Events. **JAVA SCRIPT OBJECTS –** Introduction to Object Technology, Math Object, String Object, Date Object, Boolean and Number Object, Document and Window Objects, Using Cookies.

3.1 ARRAYS

3.1.1 Introduction

Q1. What is array in javascript?

Ans :

Javascript Array is a global object which contains a list of elements. It is similar to other variables where they hold any type of data according to data type declaration but the difference is Array can hold more than one item at a time. Javascript allows a declaration of an array in many ways. Most important and common among those are 'Using array constructor' and 'Using literal notation'.

Syntax:

The array takes a list of items separated by a comma and enclosed in square brackets.

var <array_name> = [element0, element1, element2, element3, …..];

Elements inside an array can be of any data type, like string, numbers, Boolean, objects, etc which means array can consist of a string data type in its first position, number in the second, Boolean value in third and so on. Arrays in javascript are starting from index 0 and hence known as zero-based.

Here, we can see that index 0 and index 1 have string values whereas index 2 has numbers.

3.1.2 Declaring and Allocating Arrays

Q2. Explain how to create and use array in Javascript.

Ans : **(Imp.)**

Create an Array

You can create an array using two ways:

1. Using an array literal

The easiest way to create an array is by using an array literal []. For example,

```
const array1 = ["eat", "sleep"];
```

2. Using the new keyword

You can also create an array using JavaScript's new keyword.

```
const array2 = newArray("eat", "sleep");
```

In both of the above examples, we have created an array having two elements.

```
// empty array
const myList = [ ];
// array of numbers
const numberArray = [ 2, 4, 6, 8];
// array of strings
const stringArray = [ 'eat', 'work', 'sleep'];
// array with mixed data types
const newData = ['work', 'exercise', 1, true];
```

You can also store arrays, functions and other objects inside an array. For example,

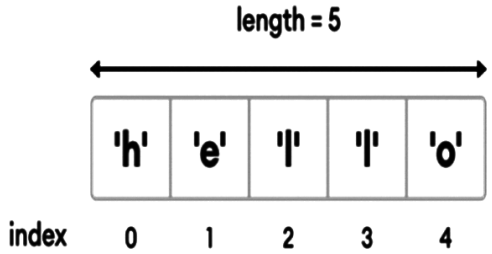
```
const newData = [
    {'task1': 'exercise'},
    [1, 2 ,3],
    functionhello() { console.log('hello')}
    ];
```

Access Elements of an Array

You can access elements of an array using indices (0, 1, 2 …). For example,

```
const myArray = ['h', 'e', 'l', 'l', 'o'];
```

```
// first element
console.log(myArray[0]);  // "h"
// second element
console.log(myArray[1]); // "e"
```



Array indexing in JavaScript

Add an Element to an Array

You can use the built-in method push() and unshift() to add elements to an array.

The push() method adds an element at the end of the array. For example,

```
let dailyActivities = ['eat', 'sleep'];
// add an element at the end
dailyActivities.push('exercise');
console.log(dailyActivities);
//  ['eat', 'sleep', 'exercise']
```

The unshift() method adds an element at the beginning of the array. For example,

```
let dailyActivities = ['eat', 'sleep'];
//add an element at the start
dailyActivities.unshift('work');
console.log(dailyActivities);
// ['work', 'eat', 'sleep']
```

Change the Elements of an Array

You can also add elements or change the elements by accessing the index value.

```
let dailyActivities = [ 'eat', 'sleep'];
// this will add the new element 'exercise' at the 2 index
dailyActivities[2] = 'exercise';
console.log(dailyActivities);
// ['eat', 'sleep', 'exercise']
```

Run Code

Suppose, an array has two elements. If you try to add an element at index 3 (fourth element), the third element will be undefined. For example,

```
let dailyActivities = [ 'eat', 'sleep'];
// this will add the new element 'exercise' at the 3 index
dailyActivities[3] = 'exercise';
console.log(dailyActivities); // ["eat", "sleep", undefined, "exercise"]
```

Run Code

Basically, if you try to add elements to high indices, the indices in between will have undefined value.

Remove an Element from an Array

You can use the pop() method to remove the last element from an array. The pop() method also returns the returned value. For example,

```
let dailyActivities = ['work', 'eat', 'sleep', 'exercise'];
// remove the last element
dailyActivities.pop();
console.log(dailyActivities);
// ['work', 'eat', 'sleep']
// remove the last element from
['work', 'eat', 'sleep']
const removedElement = daily Activities. pop();
//get removed element
console.log(removedElement); // 'sleep'
console.log(dailyActivities);  // ['work', 'eat']
```

If you need to remove the first element, you can use the shift() method. The shift() method removes the first element and also returns the removed element. For example,

```
let dailyActivities = ['work', 'eat', 'sleep'];
// remove the first element
dailyActivities.shift();
console.log(dailyActivities);
// ['eat', 'sleep']
```

Array length

You can find the length of an element (the number of elements in an array) using the length property.

For example,

```
const dailyActivities = [ 'eat', 'sleep'];

// this gives the total number of elements in an array console.log(dailyActivities.length); // 2
```

Array Methods

In JavaScript, there are various array methods available that makes it easier to perform useful calculations.

Some of the commonly used JavaScript array methods are:

Method	Description
concat()	joins two or more arrays and returns a result
indexOf()	searches an element of an array and returns its position
find()	returns the first value of an array element that passes a test
findIndex()	returns the first index of an array element that passes a test
forEach()	calls a function for each element
includes()	checks if an array contains a specified element
push()	aads a new element to the end of an array and returns the new length of an array
unshift()	adds a new element to the beginning of an array and returns the new length of an array
pop()	removes the last element of an array and returns the removed element
shift()	removes the first element of an array and returns the removed element
sort()	sorts the elements alphabetically in strings and in ascending order
slice()	selects the part of an array and returns the new array
splice()	removes or replaces existing elements and/or adds new elements

Example:

```
JavaScript Array Methods

let dailyActivities = ['sleep', 'work', 'exercise']

let newRoutine = ['eat'];

// sorting elements in the alphabetical order

dailyActivities.sort();

console.log(dailyActivities);

// ['exercise', 'sleep', 'work']

//finding the index position of string

const position = daily Activities. index Of ('work');

console.log(position); // 2

// slicing the array elements

const newDailyActivities = daily Activities. slice(1);

console.log(newDailyActivities);

// [ 'sleep', 'work']

// concatenating two arrays

const routine = daily Activities. concat (new Routine);

console.log(routine); // ["exercise", "sleep", "work", "eat"]
```

3.1.3 References and Reference Parameters

Q3. How Do you reference The Elements In An Array?

Ans : **(Imp.)**

It stores one value and its coordinates are indicated by the index on each element. Generally, the upper bound of an array determines the index of its first element, whereas the lower bound determines its index of its last element. The indexes of an array start with zero in default mode.

As part of Javascript implementation, all objects and arrays are stored in a reference system.

The Array object is used to store multiple values in a single variable:

```
const cars = ["Saab", "Volvo", "BMW"];
```

Array indexes are zero-based: The first element in the array is 0, the second is 1, and so on.

JavaScript Array Properties

Property	Description
constructor	Returns the function that created the Array object's prototype
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to an Array object

3.1.4 Passing Arrays to Functions

Q4. How to pass an array as a function parameter in JavaScript ?

Ans :

Method 1: Using the apply() method

The apply() method is used to call a function with the given arguments as an array or array-like object. It contains two parameters. The *this* value provides a call to the function and the arguments array contains the array of arguments to be passed.

The apply() method is used on the function that has to be passed as the arguments array. The first parameter is specified as 'null' and the second parameter is specified with the arguments array. This will call the function with the specified arguments array.

Syntax:

```
arrayToPass = [1, "Two", 3];
unmodifiableFunction.apply(null, arrayToPass);
```

Example

```
<!DOCTYPE html>
<html>
<head>
    <title>
        How to pass an array as a function
        parameter in JavaScript ?
    </title>
</head>
<body>
    <h1 style="color: green">
        GeeksforGeeks
    </h1>
    <b>
        JavaScript | Passing an array
        as a function parameter.
    </b>
    <p>
        The arguments passed
        are '1, "Two", 3'
    </p>
    <button onclick="passToFunction()">
        Pass to function
    </button>
    <script type="text/javascript">
        function passToFunction() {
            arrayToPass = [1, "Two", 3];
            unmodifiableFunction.apply(null,
            arrayToPass);
        }
    function unmodifiableFunction(a, b, c) {
        console.log("First value is: ", a);
        console.log("Second value is: ", b);
        console.log("Third value is: ", c);
    }
    </script>
</body>
</html>
```

Method 2: Using the Spread Syntax

The spread syntax is used in place where zero or more arguments are expected. It can be used with iterators that expands in place where there may not be a fixed number of expected arguments (like function parameters).

The required function is called as given the arguments array using the spread syntax so that it would fill in the arguments of the function from the array.

Syntax

```
arrayToPass = [1, "Two", 3];
unmodifiableFunction(...arrayToPass);
```

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>
        How to pass an array as a function
        parameter in JavaScript ?
    </title>
</head>
  <body>
    <h1 style="color: green">
        GeeksforGeeks
    </h1>
          <b>
        JavaScript | Passing an array
        as a function parameter.
    </b>
          <p>
        The arguments passed
        are '1, "Two", 3'
    </p>
     <button onclick="passToFunction()">
        Pass to function
    </button>
          <script type="text/javascript">
            function passToFunction() {
          arrayToPass = [1, "Two", 3];
        unmodifiableFunction(...arrayToPass);
        }
    function unmodifiableFunction(a, b, c)
    {
        console.log("First value is: ", a);
        console.log("Second value is: ", b);
        console.log("Third value is: ", c);
    }
    </script>
    </body>
    </html>
```

3.1.5 Multidimensional Arrays

Q5. Explain how to create and access the multi dimensional arrays in Javascript.

Ans : **(Imp.)**

A multidimensional array is an array that contains another array. For example,

```
// multidimensional array
const data = [[1, 2, 3], [1, 3, 4], [4, 5, 6]];
```

Create a Multidimensional Array

Here is how you can create multidimensional arrays in JavaScript.

Example 1

```
let studentsData = [['Jack', 24], ['Sara', 23], ['Peter', 24]];
```

Example 2

```
let student1 = ['Jack', 24];
let student2 = ['Sara', 23];
let student3 = ['Peter', 24];
// multidimensional array
let studentsData = [student1, student2, student3];
```

Here, both example 1 and example 2 creates a multidimensional array with the same data.

Access Elements of an Array

You can access the elements of a multidimensional array using indices **(0, 1, 2 ...)**. For example,

```
let x = [
['Jack', 24],
['Sara', 23],
['Peter', 24]
];
// access the first item
console.log(x[0]); // ["Jack", 24]

// access the first item of the first inner array
console.log(x[0][0]); // Jack

// access the second item of the third inner
  array
console.log(x[2][1]); // 24
```

You can think of a multidimensional array (in this case, x), as a table with 3 rows and 2 columns.

Accessing multidimensional array elements.

	Column 1	Column 2
Row 1	Jack x[0][0]	24 x[0][1]
Row 2	Sara x[1][0]	23 x[1][1]
Row 3	Peter x[2][0]	24 x[2][1]

Add an Element to a Multidimensional Array

You can use the Array's push() method or an indexing notation to add elements to a multidimensional array.

Adding Element to the Outer Array

```
let studentsData = [['Jack', 24], ['Sara', 23],];
studentsData.push(['Peter', 24]);
console.log(studentsData); //[["Jack", 24],
["Sara", 23], ["Peter", 24]
```

Adding Element to the Inner Array

```
// using index notation
let studentsData = [['Jack', 24], ['Sara', 23],];
studentsData[1][2] = 'hello';
console.log(studentsData); // [["Jack", 24],
["Sara", 23, "hello"]]

// using push()
let studentsData = [['Jack', 24], ['Sara', 23],];
studentsData[1].push('hello');
console.log(studentsData); // [["Jack", 24],
["Sara", 23, "hello"]]
```

You can also use the Array's splice() method to add an element at a specified index. For example,

```
let studentsData = [['Jack', 24], ['Sara', 23],];
// adding element at 1 index
studentsData.splice(1, 0, ['Peter', 24]);
console.log(studentsData); // [["Jack", 24],
["Peter", 24], ["Sara", 23]]
```

Remove an Element from a Multidimensional Array

You can use the Array's pop() method to remove the element from a multidimensional array. For example,

Remove Element from Outer Array

```
// remove the array element from outer array
let studentsData = [['Jack', 24], ['Sara', 23],];
studentsData.pop();
console.log(studentsData); // [["Jack", 24]]
```

Remove Element from Inner Array

```
// remove the element from the inner array
let studentsData = [['Jack', 24], ['Sara', 23]];
studentsData[1].pop();
console.log(studentsData);
// [["Jack", 24], ["Sara"]]
```

You can also use the splice() method to remove an element at a specified index. For example,

```
let studentsData = [['Jack', 24], ['Sara', 23],];
// removing 1 index array item
studentsData.splice(1,1);
console.log(studentsData); // [["Jack", 24]]
```

Iterating over Multidimensional Array

You can iterate over a multidimensional array using the Array's forEach() method to iterate over the multidimensional array. For example,

```
let studentsData = [['Jack', 24], ['Sara', 23],];
// iterating over the studentsData
    studentsData.forEach((student) => {
student.forEach((data) => {
  console.log(data);
});
});
```

Output

```
Jack
24
Sara
23
```

The first forEach() method is used to iterate over the outer array elements and the second forEach() is used to iterate over the inner array elements.

You can also use the for...of loop to iterate over the multidimensional array. For example,

```
let studentsData = [['Jack', 24], ['Sara', 23],];
for (let i of studentsData) {
for (let j of i) {
console.log(j);
}
}
```

3.2 EVENTS

3.2.1 Registering Event Handling

Q6. Write briefly about event handling in java script.

Ans : **(Imp.)**

Events are actions performed by the user such as clicking a button or hovering the mouse over a link. There are several events performed on an HTML page. You can override all of these events and use a custom JavaScript function to display feedback to the user. We've discussed some basic event handling, but you can also register event handlers with the JavaScript engine to handle more complex actions. These events can perform a simple change such as switching colors in a paragraph or returning values to the user depending on the element clicked.

Registering Event Handlers

You can also register event handlers when you have more complex functions you want to hook to your HTML elements. The advantage of registering event handlers within the JavaScript code is that we can use the "this" statement. Instead of passing a value or element to the JavaScript function like we did in previous examples, we can leave out parameters and use "this" to manipulate properties for the element.

Registering events and hooking them to functions only takes one line of code. The following code registers and event to a button named "myButton."

```
myButton.onclick = ChangeColor;
```

Now, you define the function named "Change Color." Notice that you don't add the parenthesis to the button event handler registration. This is a part of the event handler registration process in JavaScript and other languages. If you add the parenthesis, the JavaScript engine things you are trying to run a function and assign a value to the onclick method.

The following code defines a JavaScript function that changes the background color for the button.

```
function ChangeColor() {
 this.style.backgroundColor = "#000000";
}
```

When a user clicks the button named myButton, the button's background color is changed to black, which is the color that corresponds with the hexadecimal value "#000000." Most designers need to look up color values when they assign font or background colors, but the two basic ones are black and white ("#FFFFFF"). You'll also notice that we used the "this" definition, which indicates that the button clicked is the one we want to change.

You can also use this type of event registration with multiple HTML elements. Using "this," you can change styles and properties without passing the element's name to the event handler. Take the following code as an example.

```
myButton.onclick = ChangeColor;
myOtherButton.onclick = ChangeColor;
```

The above code defines two buttons named "myButton" and "myOtherButton" and assigns the same event handler to the "onclick"button. Using the same function code, the "this" assignment points the function to the currently clicked button. This type of coding makes your code more dynamic and efficient, so you don't need to specify the button that is clicked for each element on your page.

Types of Event Handlers

We've focused mostly on the "onclick" event handler, but there are several more that you'll work with when you code in JavaScript. This section will take a look at several of the other handlers available within the HTML code.

First, there is the "onload" and "unload" events. These two events trigger when a web page is loaded or unloaded in the browser. Usually, coders use the onload event to fire a JavaScript function. For instance, you might want to register events after a page loads in the browser. We've already created our function to change an element's background color, but we can call a function that registers each event when a page loads to make the website's code more efficient. Here is an example of a web page using the onload event.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<script>
function RegisterEvents ( )
{
    clickme.onclick = ChangeStyle;
    username.onclick = ChangeStyle;
}
    function ChangeStyle ( )
{
this.style.backgroundColor = "#000000";
}
</script>
</head>
<body onload="RegisterEvents()" >
<div id="username"> Hi, user! </div>
<button id="clickme"> Click Me! </button>
</body>
</html>
```

The above code has two elements in its HTML. The first is a div named "username" and the second is a button named "clickme." We used the previous "ChangeStyle" function to register an event handler for the onclick action. Notice the "body" tag has the "onload" event added to its property definition. This event fires when the page loads, which then calls the function "RegisterEvents" to register event handlers for the two elements. When a user clicks these elements, the background color is changed to black ("#000000"). You can also use the "unload" event in the body tag to run functions when the user leaves a page or unloads it in the browser.

There are four events that occur with a user mouse: onmouseover, onmouseout, onmouse down, onmouseup. The first two occur when the user hovers over an element and then moves the mouse away from the object. The second two occur when the user clicks the mouse button down and then release the button. When you change an attribute for the onmousedown event, you also need to change the style back to the original when the user releases the mouse button (onmouseup). The same is true for the onmouseover and onmouseout events.

3.2.2 Event Onload

Q7. Explain onload event in javascript with example.

Ans : **(Imp.)**

In JavaScript, this event can apply to launch a particular function when the page is fully displayed. It can also be used to verify the type and version of the visitor's browser. We can check what cookies a page uses by using the **onload** attribute.

In HTML, the onload attribute fires when an object has been loaded. The purpose of this attribute is to execute a script when the associated element loads.

In HTML, the **onload** attribute is generally used with the **<body>** element to execute a script once the content (including CSS files, images, scripts, etc.) of the webpage is completely loaded. It is not necessary to use it only with <body> tag, as it can be used with other HTML elements.

The difference between the document. onload and window.onload is: document. onload triggers before the loading of images and other external content. It is fired before the window.onload. While the window.onload triggers when the entire page loads, including CSS files, script files, images, etc.

Syntax

window.onload = fun()

Let's understand this event by using some examples.

Example1

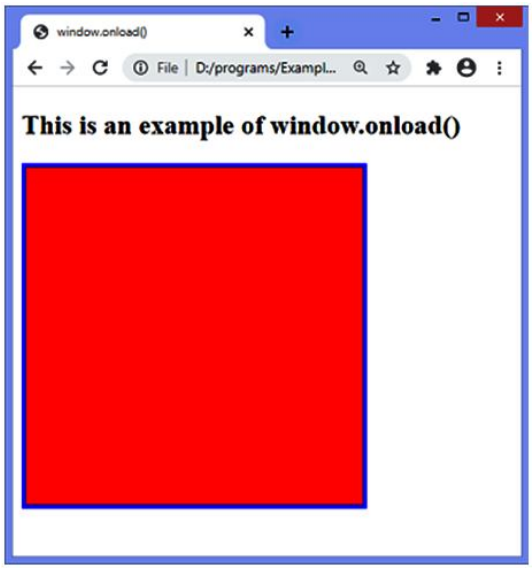
In this example, there is a div element with a height of 200px and a width of 200px. Here, we are using the **window.onload()** event change the background color, width, and height of the **div** element after loading the web page.

The background color is set to **'red'**, and width and height are set to **300px** each.

```
<!DOCTYPE html>
<html>
<head>
<meta charset = " utf-8">
<title> window.onload() </title>
<style type = "text/css">
#bg{
width: 200px;
height: 200px;
border: 4px solid blue;
}
</style>
<script type = "text/javascript">
window.onload = function(){
document.getElementById("bg").style.
backgroundColor = "red";
document.getElementById("bg").style.
width = "300px";
document.getElementById("bg").style.
height = "300px";
}
</script>
</head>
<body>
<h2> This is an example of window.
onload() </h2>
<div id = "bg"></div>
</body>
</html>
```

Output

After the execution of the code and loading of the page, the output will be -



3.2.3 On Mouseover

Q8. Explain onmouseover event in javascript.

Ans :

The mouseover event occurs when a mouse pointer comes over an element, and mouseout – when it leaves.

These events are special, because they have property relatedTarget. This property complements target. When a mouse leaves one element for another, one of them becomes target, and the other one – relatedTarget.

For mouseover

- Event.target: is the element where the mouse came over.
- Event.relatedTarget: is the element from which the mouse came (relatedTarget '! target).

For mouseout the reverse

- **Event.target:** is the element that the mouse left.
- **Event.relatedTarget:** is the new under-the-pointer element, that mouse left for (target '! relatedTarget).

Example

You can try to run the following code to learn how to work with *onmouseover* event in JavaScript "

```
<html>
   <head>
      <script>
         <!—
            function sayHello(){
               alert("Mouse Over")
            }
         //—>
      </script>
   </head>
   <body>
      <ponmouseover="sayHello()">This is demo text for mouseover event.</p>
   </body>
</html>
```

3.2.4 On Mouseout

Q9. Explain onmouseout event in Javascript

Ans :

The onmouseout event occurs when the mouse pointer is moved out of an element, or out of one of its children.

Syntax:

- **In HTML:**

  <element onmouseout="myScript">

- **In JavaScript:**

  object.onmouseout = function(){myScript};

- **In JavaScript, using the addEvent Listener() method:**

  object.addEventListener("mouseout", myScript);

Example: Using the addEventListener() method

```
<!DOCTYPE html>
<html>
<head>
   <title>DOM onmouseout Event</title>
</head>
<body>
   <center>
      <h1 id="hID">
         GeeksforGeeks
      </h1>
      <h2>DOM onmouseout Event</h2>
      <script>
         document.getElementById(
            "hID").addEventListener(
            "mouseover", over);
         document.getElementById(
```

```
            "hID").addEventListener(
            "mouseout", out);
        function over() {
            document.getElementById(
            "hID").style.color = "green";
        }
        function out() {
            document.getElementById(
            "hID").style.color = "black";
        }
    </script>
</center>
</body>
</html>
```

3.2.5 Onfocus

Q10. Explain onfocus event in javascript.

Ans :

The onfocus event occurs when an element gets focus.

The onfocus event is most often used with <input>, <select>, and <a>.

This attribute is commonly used with elements such as <input>, <a>, <select>. This event attribute is sustained by every HTML element except the ones like <base>, <head>, <bdo>,
, <html>, <meta>, <iframe>, <param>, <style>, <script>, and <title> elements. This event handler executes when an element becomes focus on the screen of user. The elements focus can be when the user clicks on a text box. At that time, it is in focus by the user, since the person has clicked on it.

Syntax

Given below is the syntax of the onfocus event handler:

```
<element onfocus = "script">
```

Here, the attribute value is the script value that runs when the attribute onfocus is called.

Example #1

JavaScript program that changes the color of text field when it is in focus.

```
<!DOCTYPE html>
<html>
<body>
<p>Sample for onfocus event handler</p>
User Name
<input type="text" id="fname" onfocus
="func(this.id)"><br>
Confirm user name: <input type="text"
id="lname" onfocus="func(this.id)">
<script>
function func(x) {
document.getElementById(x).style.
background = "red";
}
</script>
</body>
</html>
```

Output

On executing the code, two text fields will be displayed.

Sample for onfocus event handler

User Name:
Confirm user name:

On clicking the second text box also, the color of it changes to red.

Sample for onfocus event handler

User Name:
Confirm user name:

3.2.6 ONBLUR

Q11. Explain onblur event in javascript with example.

Ans : **(Imp.)**

onblur is an in-built event that is available in JavaScript which gets triggered when the elements

on which it is applied loses its focus. This is one of the significant events from the events which are supported by JavaScript. This event is fired when the item comes out of focus on the web page. onblur event is mostly used with the input field element such as in form where input validation can be performed for example when the user enters input into the field and goes to the next element, the onblur event can be attached on that field and validation can be performed.

Syntax:

We can attach a script to execute whenever the onblur event occurs in the following ways.

HTML

```
<element onblur = "functionName()" >
```

JavaScript

```
elementObj.onblur = function () { //script };
```

Using addEventListener() method in JavaScript:

```
elementObj.addEventListener( "blur" , function() { //script } );
```

Example #1 – In HTML

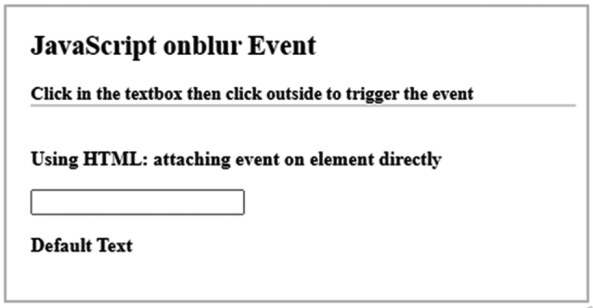
Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset = "UTF-8">
<title>
JavaScript onblur Event
</title>
<style>
.body-data {
border : #81D4FA 2px solid;
background-color : #03a9f400;
text-align : left;
padding-left : 20px;
padding-bottom: 20px;
height : auto;
width : auto;
}
.resultText {
margin: 0 0 3px 0;
padding: 0px;
display: block;
font-weight: bold;
}
.heading {
font-weight: bold;
border-bottom: 2px solid #ddd;
font-size: 15px;
width: 98%;
}
</style>
</head>
<body>
<div class = "body-data" >
<div class = "heading" >
<h2> JavaScript onblur Event </h2>
<span> Click in the textbox then click outside to trigger the event </span>
</div>
<div class = "resultText" >
</br>
<p> Using HTML: attaching event on element directly </p>
<!— attaching mousedown event on button —>
<input type = "text" id = "myText" onblur = "fireEvent()" >
<p id = "result1" > Default Text </p>
</div>
</div>
<script type = "text/javascript">
// This function will be called whenever onblur event occurs
```

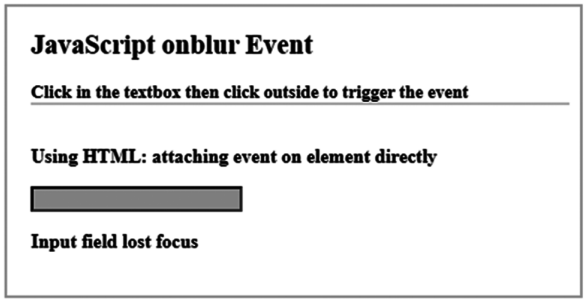
```
function fireEvent( ) {
    document.getElementById( "myText" ).style.backgroundColor = '#81D4FA';
    document.getElementById("result1").innerHTML = " Input field lost focus ";
}
</script>
</body>
</html>
```

Output

Before the event occurs:



After the event occurs



3.2.7 ONSUBMIT

Q12. Explain onsubmit event in javascript with example.

Ans : **(Imp.)**

The JavaScript onsubmit is one of the event handling function used for performing the operations in web based applications. It is used to validate the datas and whatever the client user is requested to the server it is to be validated and mainly used in the html form controls also it is used to set the html contents of the some input like hidden elements is to be validated before the request is submitting to the server side. If suppose the user is forget to submit the required fields in the form they can be cancelled the submission within the onsubmit event.

Syntax

Every event handling function must have their own syntax and rules in the JavaScript code.

```
<html>
<head>
<script>
function functionName()
{
    — some javascript logic codes —
}
</script>
</head>
<body>
<form id="" method="" action="" onsubmit=" functionName()">
— some html input tags and elements —
</form>
</body>
</html>
```

The above code is the basic syntax for using onsubmit().

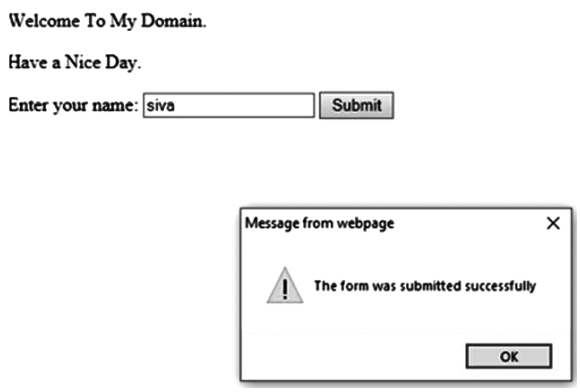
Example #1

Code:

```
<!DOCTYPEhtml>
<html>
<body>
<p>Welcome To My Domain.</p>
<p>Have a Nice Day.</p>
<formid="first"action="first.jsp">
Enter your name: <inputtype="text" name=" fname">
<inputtype="submit"value="Submit">
</form>
<script>
```

```
document.getElementById("first"). onsubmit
= function(){demo()};
functiondemo() {
alert("The form was submitted successfully");
}
</script>
</body>
</html>
```

Output:



3.2.8 ONRESET

Q13. Explain onreset event in javascript with example.

Ans :

The onreset event in HTML DOM occurs when a form is reset. Only <form> tag is supported in this event.

Syntax:

- **In HTML**

```
<element onreset="myScript">
```

- **In JavaScript**

```
object.onreset = function(){myScript};
```

- **In JavaScript, using the addEvent Listener() method:**

```
object.addEventListener("reset", myScript);
```

Example

Using the addEventListener() method

```
<!DOCTYPE html>
<html>  <head>
    <title>
      HTML DOM onreset Event
  </title>
</head>

<body>
    <center>
        <h1 style="color:green">
          GeeksforGeeks
      </h1>
        <h2>HTML DOM onreset Event</h2>

        <form id="formID">
            Email:
            <input type="email">
            <input type="submit"
                value="Submit">
            <input type="reset">
        </form>
    </center>
    <script>
        document.getElementById(
          "formID").addEventListener("reset",
                GFGfun);
        function GFGfun() {
            alert("Form reset");
        }
    </script>
</body></html>
```

3.2.9 Event Bubbling

Q14. What is event bubbling in java script? Explain with an example.

Ans : **(Imp.)**

Event Bubbling

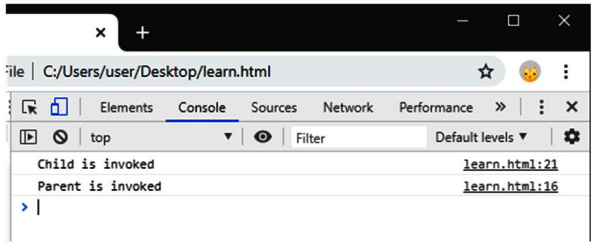
While developing a webpage or a website via JavaScript, the concept of event bubbling is used where the event handlers are invoked when one element is nested on to the other element and are

part of the same event. This technique or method is known as Event Bubbling. Thus, while performing event flow for a web page, event bubbling is used. We can understand event bubbling as a sequence of calling the event handlers when one element is nested in another element, and both the elements have registered listeners for the same event. So beginning from the deepest element to its parents covering all its ancestors on the way to top to bottom, calling is performed.

Example of Event Bubbling

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content ="width
    =device-width">
    <title>Event Bubbling</title>
</head>
<body>
    <div id="p1">
    <button id="c1">I am child button </
    button>
    </div>
    <script>
    var parent = document.query Selector
                ('#p1');
    parent.addEventListener('click', function
                ()){
        console.log("Parent is invoked");
    });
    var child = document.query Selector
                ('#c1');
    child.addEventListener('click', function (){
        console.log("Child is invoked");
    });
  </script>
</body>
</html>
```

Output

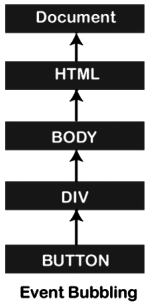


Explanation of Code

- The above code is a HTML and JavaScript based code.
- We have used a div tag having div id = p1 and within div we have nested a button having button id = c1.
- Now, within the JavaScript section, we have assigned the html elements (p1 and c1) using the querySelector () function to the variable parent and child.
- After that, we have created and included an event which is the click event to both div element and child button. Also created two functions that will help us to know the sequence order of the execution of the parent and child. It means if the child event is invoked first, "child is invoked" will be printed otherwise "parent is invoked" will get printed.
- Thus, when the button is clicked, it will first print "child is invoked" which means that the function within the child event handler executes first. Then it moves to the invocation of the div parent function.

The sequence has taken place due to the concept of event bubbling. Thus, in this way event bubbling takes place.

We can also understand the flow of event with the help of the below flow chart:



Event Bubbling

It means when the user click on the button, the click event flows in this order from bottom to top.

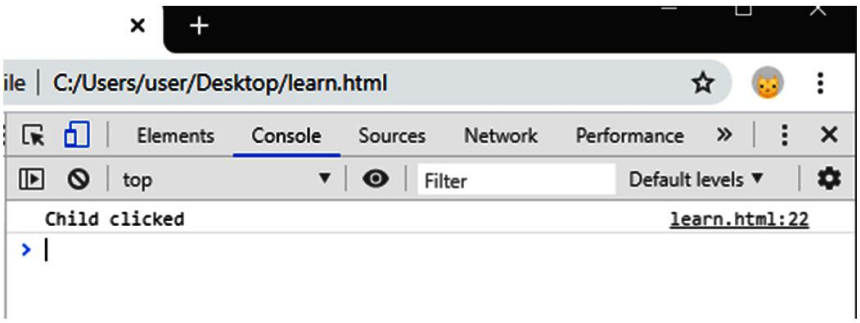
Stopping Bubbling

Beginning from the target and moving towards the top is the bubbling i.e. starting from the child to its parent, it moves straight upwards. But a handler can also take decision to stop the bubbling when the event has been processed completely. In JavaScript, we use the **event.stop Propagation ()** method.

For example:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
    <title>Event Bubbling</title>
</head>
<body>
    <div id="p1">
        <button id="c1" onclick="event.stopPropagation()">I am child</button>
    </div>
        <script>
    var parent = document.querySelector('#p1');
    parent.addEventListener('click', function(){
    console.log("Parent is invoked");
    });
    var child = document.querySelector('#c1');
    child.addEventListener('click', function(){
    console.log("Child is invoked");
 });
</script>
</body>
</html>
```

In the above code, when we click on the button, it will not work because event.stopPropagation () method is being invoked here due to which th parent function will not be invoked.



3.2.10 MORE EVENTS

Q15. Explain Javascript onclick event.

Ans : **(Imp.)**

JavaScript onclick event

The **onclick** event generally occurs when the user clicks on an element. It allows the programmer to execute a JavaScript's function when an element gets clicked. This event can be used for validating a form, warning messages and many more.

Using JavaScript, this event can be dynamically added to any element. It supports all HTML elements except **<html>**, **<head>**, **<title>**, **<style>**, **<script>**, **<base>**, **<iframe>**, **<bdo>**, **
, **<meta>, and **<param>**. It means we cannot apply the **onclick** event on the given tags.

In HTML, we can use the **onclick** attribute and assign a JavaScript function to it. We can also use the JavaScript's **addEventListener()** method and pass a **click** event to it for greater flexibility.

Syntax

Now, we see the syntax of using the **onclick** event in HTML and in javascript (without **addEventListener()** method or by using the **addEventListener()** method).

In HTML

<element onclick = "fun()"**>**

In JavaScript

object.onclick = function() { myScript };

In JavaScript by using the addEventListener() method

object.addEventListener("click", myScript);

Example

In this example, we are using JavaScript's **onclick** event. Here we are using the **onclick** event with the paragraph element.

When the user clicks on the paragraph element, the corresponding function will get executed, and the text of the paragraph gets changed. On clicking the **<p>** element, the background color, size, border, and color of the text will also get change.

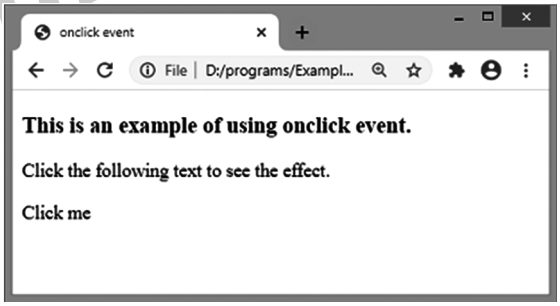
<!DOCTYPE html**>**

<html>

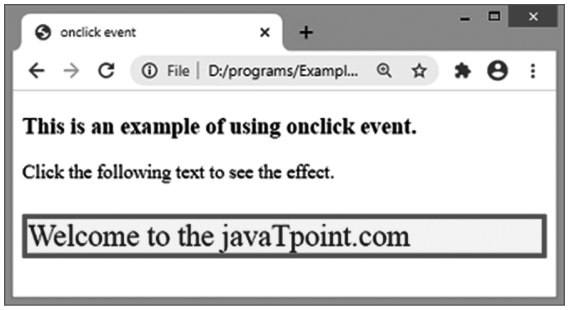
<head>

```
<title> onclick event </title>
</head>
<body>
<h3> This is an example of using onclick event. </h3>
<p> Click the following text to see the effect. </p>
<p id = "para">Click me</p>
<script>
document.getElementById("para").onclick = function() {
fun()
};
function fun() {
document.getElementById("para").innerHTML = "Welcome to the javaTpoint.com";
document.getElementById("para").style.color = "blue";
document.getElementById("para").style.backgroundColor = "yellow";
document.getElementById("para").style.fontSize = "25px";
document.getElementById("para").style.border = "4px solid red";
}
</script>
</body> </html>
```

Output



After clicking the text ***Click me,*** the output will be -



Q16. Explain Javascript dblclick event.

Ans :

JavaScript dblclick event

The dblclick event generates an event on double click the element. The event fires when an element is clicked twice in a very short span of time. We can also use the JavaScript's addEvent Listener() method to fire the double click event.

In HTML, we can use the ondblclick attri-bute to create a double click event.

Syntax

Now, we see the syntax of creating double click event in HTML and in javascript (without using addEventListener() method or by using the addEventListener() method).

In HTML

<element ondblclick = "fun()"**>**

In JavaScript

object.ondblclick = function() {myScript};

In JavaScript by using the addEventListener() method

object.addEventListener("dblclick", my Script);

Example

<!DOCTYPE html**>**

<html>

<head>

</head>

<body>

<h1 id = "heading"**>** Hello world :):) **</h1>**

<h2> Double Click the text "Hello world" to see the effect. **</h2>**

** ** This is an example of creating the double click event using JavaScript. ** **

<script>

document.getElementById("heading").ondblclick = function() { fun() };

function fun()

{ document.getElementById ("heading").innerHTML = " Welcome to the javaTpoint.com ";

}

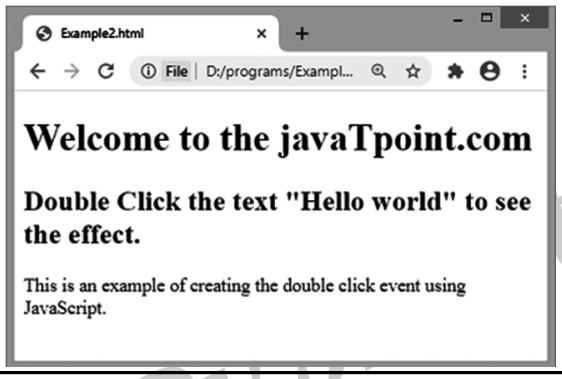
</script>

</body>

</html>

Output

After double-clicking the text **"Hello world"**, the output will be -



Q17. Explain JavaScript onresize event.

Ans : **(Imp.)**

JavaScript onresize event

The onresize event in JavaScript generally occurs when the window has been resized. To get the size of the window, we can use the JavaScript's window.outer Width and window.outer Height events. We can also use the JavaScript's properties such as inner Width, innerHeight, clientWidth, ClientHeight, offsetWidth, offsetHeight to get the size of an element.

In HTML, we can use the onresize attribute and assign a JavaScript function to it. We can also use the JavaScript's addEventListener() method and pass a resize event to it for greater flexibility.

Syntax

Now, we see the syntax of using the **onresize** event in HTML and in javascript (without **addEventListener()** method or by using the **addEventListener()** method).

In HTML

<element onresize = "fun()"**>**

In JavaScript

object.onresize = function() { myScript };

In JavaScript by using the addEventListener() method

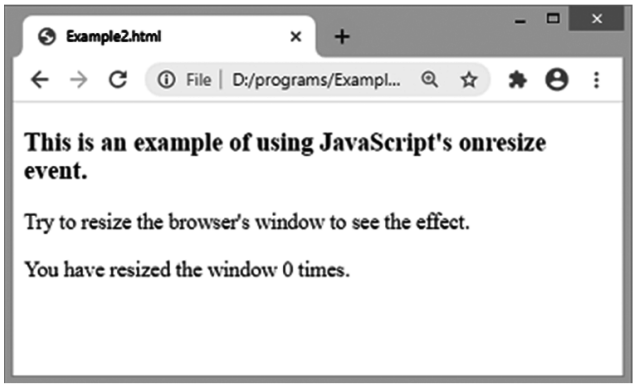
object.addEventListener("resize",my Script);

Example

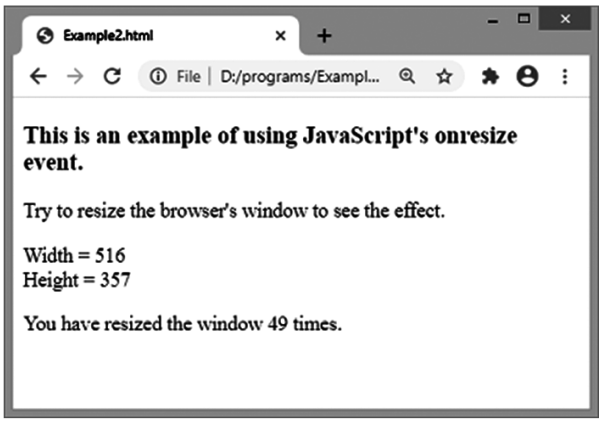
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h3> This is an example of using JavaScript's onresize event. </h3>
<p> Try to resize the browser's window to see the effect. </p>
<p id = "para"> </p>
<p> You have resized the window <span id = "s1"> 0 </span> times.</p>
<script>
document.getElementsByTagName("BODY")[0].onresize = function() {fun()};
var i = 0;
function fun() {
var res = "Width = " + window.outerWidth + "<br>" + "Height = " + window.outer Height;
document.getElementById("para").innerHTML = res;
    var res1 = i += 1;
document.getElementById("s1").innerHTML = res1;
}
</script>
</body>
</html>
```

Output

After the execution of the above code, the output will be -



When we try to resize the window, the output will be -



3.3 JAVA SCRIPT OBJECTS

3.3.1 Introduction to Object Technology

Q18. Explain about Javascript objects with example.

Ans : **(Imp.)**

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

1. JavaScript Object by object literal

The syntax of creating object using object literal is given below:

object={property1:value1, property2: value2..... property N: valueN}

As you can see, property and value is separated by : (colon).

Let's see the simple example of creating object in JavaScript.

```
<script>
emp={id:102,name: "Shyam Kumar", salary: 40000}

document.write(emp.id+" "+ emp.name +" "+ emp.salary);
</script>
```

Output of the above example

102 Shyam Kumar 40000

2. By creating instance of Object

The syntax of creating object directly is given below:

var objectname=new Object();

Here, **new keyword** is used to create object.

Let's see the example of creating object directly.

```
<script>
var emp = new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write (emp.id+ " "+ emp.name +" "+ emp.salary);
</script>
```

Output of the above example

101 Ravi 50000

3. By using an Object constructor

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

The **this keyword** refers to the current object.

The example of creating object by object constructor is given below.

```
<script>
```

```
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal", 30000);
document . write (e.id+" "+e.name+" " + e.salary);
</script>
```

Output of the above example

103 Vimal Jaiswal 30000

Defining method in JavaScript Object

We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
this.changeSalary=changeSalary;
function changeSalary(otherSalary){
this.salary=otherSalary;
}
}
e=new emp(103,"Sonoo Jaiswal", 30000);
document.write(e.id+" "+e.name+" "+e.salary);
e.changeSalary(45000);
document.write ("<br>"+e.id+" "+ e.name +" " + e.salary);
</script>
```

Output of the above example

103 Sonoo Jaiswal 30000

103 Sonoo Jaiswal 45000

Sl. No.	JavaScript	Object Methods
1.	Object.assign()	➢ This method is used to copy enumerable and own properties from a source object to a target object
2.	Object.create()	➢ This method is used to create a new object with the specified prototype object and properties.
3.	Object.defineProperty()	➢ This method is used to describe some behavioral attributes of the property.
4.	Object.defineProperties()	➢ This method is used to create or configure multiple object properties.
5.	Object.entries()	➢ This method returns an array with arrays of the key, value pairs.
6.	Object.freeze()	➢ This method prevents existing properties from being removed.
7.	Object.getOwnPropertyDescriptor()	➢ This method returns a property descriptor for the specified property of the specified object.
8.	Object.getOwnPropertyDescriptors()	➢ This method returns all own property descriptors of a given object.
9.	Object.getOwnPropertyNames()	➢ This method returns an array of all properties (enumerable or not) found.
10.	Object.getOwnPropertySymbols()	➢ This method returns an array of all own symbol key properties.
11.	Object.getPrototypeOf()	➢ This method returns the prototype of the specified object.
12.	Object.is()	➢ This method determines whether two values are the same value.
13.	Object.isExtensible()	➢ This method determines if an object is extensible
14.	Object.isFrozen()	➢ This method determines if an object was frozen.
15.	Object.isSealed()	➢ This method determines if an object is sealed.
20.	Object.values()	➢ This method returns an array of values.

3.3.2 Math Object

Q19. Explain about math object in javascript.

Ans : **(Imp.)**

The **JavaScript math** object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors.

JavaScript Math Methods

Methods	Description
abs()	It returns the absolute value of the given number.
acos()	It returns the arccosine of the given number in radians.
asin()	It returns the arcsine of the given number in radians.
atan()	It returns the arc-tangent of the given number in radians.
cbrt()	It returns the cube root of the given number.
ceil()	It returns a smallest integer value, greater than or equal to the given number.
cos()	It returns the cosine of the given number.
cosh()	It returns the hyperbolic cosine of the given number.
exp()	It returns the exponential form of the given number.
floor()	It returns largest integer value, lower than or equal to the given number.
hypot()	It returns square root of sum of the squares of given numbers.
log()	It returns natural logarithm of a number.
max()	It returns maximum value of the given numbers.
min()	It returns minimum value of the given numbers.
pow()	It returns value of base to the power of exponent.
random()	It returns random number between 0 (inclusive) and 1 (exclusive).
round()	It returns closest integer value of the given number.
sign()	It returns the sign of the given number
sin()	It returns the sine of the given number.
sinh()	It returns the hyperbolic sine of the given number.
sqrt()	It returns the square root of the given number
tan()	It returns the tangent of the given number.
tanh()	It returns the hyperbolic tangent of the given number.
trunc()	It returns an integer part of the given number.

Math.sqrt(n)

The JavaScript math.sqrt(n) method returns the square root of the given number.

```
Square Root of 17 is: <span id="p1"></span>
    <script>
document.getElementById('p1').innerHTML=Math.sqrt(17);
    </script>
```

Output

Square Root of 17 is:4.123105625617661

Math.random()

The JavaScript math.random() method returns the random number between 0 to 1.

Random Number is: **<span** id="p2"**>**

<script>

document.getElementById('p2').innerHTML=Math.random();

</script>

Output

Random Number is: 0.5280061261755884

Math.pow(m,n)

The JavaScript math.pow(m,n) method returns the m to the power of n that is m^n.

3 to the power of 4 is: **<span** id="p3"**>**

<script>

document.getElementById('p3').innerHTML=Math.pow(3,4);

</script>

Output:

3 to the power of 4 is: 81

Math.floor(n)

The JavaScript math.floor(n) method returns the lowest integer for the given number. For example 3 for 3.7, 5 for 5.9 etc.

Floor of 4.6 is: **<span** id="p4"**>**

<script>

document.getElementById('p4').innerHTML=Math.floor(4.6);

</script>

Output:

Floor of 4.6 is: 4

Math.ceil(n)

The JavaScript math.ceil(n) method returns the largest integer for the given number. For example 4 for 3.7, 6 for 5.9 etc.

Ceil of 4.6 is: **<span** id="p5"**>**

<script>

document.getElementById('p5').innerHTML=Math.ceil(4.6);

</script>

Output:

Ceil of 4.6 is: 5

Math.round(n)

The JavaScript math.round(n) method returns the rounded integer nearest for the given number. If fractional part is equal or greater than 0.5, it goes to upper value 1 otherwise lower value 0. For example 4 for 3.7, 3 for 3.3, 6 for 5.9 etc.

```
Round of 4.3 is: <span id="p6"></span><br>
Round of 4.7 is: <span id="p7"></span>
<script>
    document.getElementById('p6').innerHTML=Math.round(4.3);
    document.getElementById('p7').innerHTML=Math.round(4.7);
</script>
```

Output

Round of 4.3 is: 4

Round of 4.7 is: 5

Math.abs(n)

The JavaScript math.abs(n) method returns the absolute value for the given number. For example 4 for -4, 6.6 for -6.6 etc.

```
Absolute value of -4 is: <span id="p8"></span>
<script>
document.getElementById('p8').innerHTML=Math.abs(-4);
</script>
```

Output

Absolute value of -4 is: 4

3.3.3 String Object

Q20. Explain String object in Java script.

Ans : **(Imp.)**

The JavaScript string is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

1. By string literal

The string literal is created using double quotes. The syntax of creating string using string literal is given below:

```
var stringname="string value";
```

Let's see the simple example of creating string literal.

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

Output

This is string literal

2. By string object (using new keyword)

The syntax of creating string object using new keyword is given below:

var stringname=new String("string literal");

Here, **new keyword** is used to create instance of string.

Let's see the example of creating string in JavaScript by new keyword.

```
<script>
var stringname=new String("hello javascript string");
document.write(stringname);
</script>
```

Output

hello javascript string

JavaScript String Methods

Methods	Description
charAt()	It provides the char value present at the specified index.
charCodeAt()	It provides the Unicode value of a character present at the specified index.
concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndexOf()	It provides the position of a char value present in the given string by searching a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.
match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.
toLocaleLowerCase()	It converts the given string into lowercase letter on the basis of host?s current locale.
toUpperCase()	It converts the given string into uppercase letter.
toLocaleUpperCase()	It converts the given string into uppercase letter on the basis of host?s current locale.
toString()	It provides a string representing the particular object.
valueOf()	It provides the primitive value of string object.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.

1. JavaScript String charAt(index) Method

The JavaScript String charAt() method returns the character at the given index.

```
<script>
var str="javascript";
document.write(str.charAt(2));
</script>
```

Output

v

2. JavaScript String concat(str) Method

The JavaScript String concat(str) method concatenates or joins two strings.

```
<script>
var s1="javascript ";
var s2="concat example";
var s3=s1.concat(s2);
document.write(s3);
</script>
```

Output

javascript concat example

3. JavaScript String indexOf(str) Method

The JavaScript String indexOf(str) method returns the index position of the given string.

```
<script>
var s1="javascript from javatpoint indexof";
var n=s1.indexOf("from");
document.write(n);
</script>
```

Output

11

4. JavaScript String lastIndexOf(str) Method

The JavaScript String lastIndexOf(str) method returns the last index position of the given string.

```
<script>
var s1="javascript from javatpoint indexof";
var n=s1.lastIndexOf("java");
document.write(n);
</script>
```

Output

16

5. JavaScript String toLowerCase() Method

The JavaScript String toLowerCase() method returns the given string in lowercase letters.

```
<script>
var s1="JavaScript toLowerCase Example";
var s2=s1.toLowerCase();
document.write(s2);
</script>
```

Output

javascript tolowercase example

6. JavaScript String toUpperCase() Method

The JavaScript String toUpperCase() method returns the given string in uppercase letters.

```
<script>
var s1="JavaScript toUpperCase Example";
var s2=s1.toUpperCase();
document.write(s2);
</script>
```

Output

JAVASCRIPT TOUPPERCASE EXAMPLE

7. JavaScript String slice(beginIndex, endIndex) Method

The JavaScript String slice(beginIndex, endIndex) method returns the parts of string from given beginIndex to endIndex. In slice() method, beginIndex is inclusive and endIndex is exclusive.

```
<script>
var s1="abcdefgh";
var s2=s1.slice(2,5);
document.write(s2);
</script>
```

Output

cde

8. JavaScript String trim() Method

The JavaScript String trim() method removes leading and trailing whitespaces from the string.

```
<script>
var s1= "javascript trim";
var s2=s1.trim();
document.write(s2);
</script>
```

Output

javascript trim

3.3.4 Date Object

Q21. Explain Javascript date object.

Ans :

The **JavaScript date** object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

You can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

JavaScript	Date Methods
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.
getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.

getUTCDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of universal time.
getUTCDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of universal time.
getUTCFullYears()	It returns the integer value that represents the year on the basis of universal time.
getUTCHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of universal time.
getUTCMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of universal time.
getUTCMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of universal time.
getUTCSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of universal time.
setDate()	It sets the day value for the specified date on the basis of local time.
setDay()	It sets the particular day of the week on the basis of local time.
setFullYears()	It sets the year value for the specified date on the basis of local time.
setHours()	It sets the hour value for the specified date on the basis of local time.
setMilliseconds()	It sets the millisecond value for the specified date on the basis of local time.
setMinutes()	It sets the minute value for the specified date on the basis of local time.
setMonth()	It sets the month value for the specified date on the basis of local time.
setSeconds()	It sets the second value for the specified date on the basis of local time.
setUTCDate()	It sets the day value for the specified date on the basis of universal time.
setUTCDay()	It sets the particular day of the week on the basis of universal time.
setUTCFullYears()	It sets the year value for the specified date on the basis of universal time.
setUTCHours()	It sets the hour value for the specified date on the basis of universal time.
setUTCMilliseconds()	It sets the millisecond value for the specified date on the basis of universal time.
setUTCMinutes()	It sets the minute value for the specified date on the basis of universal time.
setUTCMonth()	It sets the month value for the specified date on the basis of universal time.
setUTCSeconds()	It sets the second value for the specified date on the basis of universal time.
toDateString()	It returns the date portion of a Date object.

toISOString()	It returns the date in the form ISO format string.
toJSON()	It returns a string representing the Date object. It also serializes the Date object during JSON serialization.
toString()	It returns the date in the form of string.
toTimeString()	It returns the time portion of a Date object.
toUTCString()	It converts the specified date in the form of string using UTC time zone.
valueOf()	It returns the primitive value of a Date object.

JavaScript Date Example

```
Current Date and Time: <span id="txt"></span>
<script>
var today=new Date();
document.getElementById('txt').innerHTML=today;
</script>
```

Output

Current Date and Time: Wed May 11 2022 16:14:52 GMT+0530 (India Standard Time)

Let's see another code to print date/month/year.

```
<script>
var date=new Date();
var day=date.getDate();
var month=date.getMonth()+1;
var year=date.getFullYear();
document.write("<br>Date is: "+day+"/"+month+"/"+year);
</script>
```

Output

Date is: 11/5/2022

JavaScript Current Time Example

```
Current Time: <span id="txt"></span>
<script>
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
</script>
```

Output

Current Time: 16:14:52

JavaScript Digital Clock Example

Let's see the simple example to display digital clock using JavaScript date object.There are two ways to set interval in JavaScript: by setTimeout() or setInterval() method.

Current Time: <span id="txt"**>**

```
<script>
window.onload=function(){getTime();}
function getTime(){
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
//    add a zero in front of numbers<10
      m=checkTime(m);
      s=checkTime(s);
      document.getElementById('txt').innerHTML=h+":"+m+":"+s;
      setTimeout(function(){getTime()},1000);
}
//setInterval("getTime()",1000);//another way
function checkTime(i){
if (i<10){
      i="0" + i;
}
return i;
}
</script>
```

Output

Current Time: 16:16:30

3.3.5 Boolean and Number Object

Q22. Explain Javascript Boolean and number object.

Ans : **(Imp.)**

JavaScript Boolean is an object that represents value in two states: true or false. You can create the JavaScript Boolean object by Boolean() constructor as given below.

```
Boolean b=new Boolean(value);
```

The default value of JavaScript Boolean object is *false*.

JavaScript Boolean Example

```
<script>
```

```
document.write(10<20);//true
document.write(10<5);//false
</script>
```

JavaScript Boolean Properties

Property	Description
Constructor	Returns the reference of Boolean function that created Boolean object.
Prototype	Enables you to add properties and methods in Boolean prototype.

JavaScript Boolean Methods

Method	Description
toSource()	returns the source of Boolean object as a string.
toString()	converts Boolean into String.
valueOf()	converts other type into Boolean.

JavaScript Number Object

The **JavaScript number** object *enables you to represent a numeric value*. It may be integer or floating-point. JavaScript number object follows IEEE standard to represent the floating-point numbers.

By the help of Number() constructor, you can create number object in JavaScript. For example:

```
var n=new Number(value);
```

If value can't be converted to number, it returns NaN(Not a Number) that can be checked by isNaN() method.

You can direct assign a number to a variable also. For example:

```
var x=102;//integer value
var y=102.7;//floating point value
var z=13e4;//exponent value, output: 130000
var n=new Number(16);//integer value by number object
```

Output

102 102.7 130000 16

JavaScript Number Constants

Constant	Description
MIN_VALUE	returns the largest minimum value.
MAX_VALUE	returns the largest maximum value.
POSITIVE_INFINITY	returns positive infinity, overflow value.
NEGATIVE_INFINITY	returns negative infinity, overflow value.
NaN	represents "Not a Number" value.

JavaScript Number Methods

Methods	Description
isFinite()	It determines whether the given value is a finite number.
isInteger()	It determines whether the given value is an integer.
parseFloat()	It converts the given string into a floating point number.
parseInt()	It converts the given string into an integer number.
toExponential()	It returns the string that represents exponential notation of the given number.
toFixed()	It returns the string that represents a number with exact digits after a decimal point.
toPrecision()	It returns the string representing a number of specified precision.
toString()	It returns the given number in the form of string.

3.3.6 Document and Window Objects

Q23. Write about Document Object Model.

Ans : **(Imp.)**

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.

As mentioned earlier, it is the object of window. So

window.document

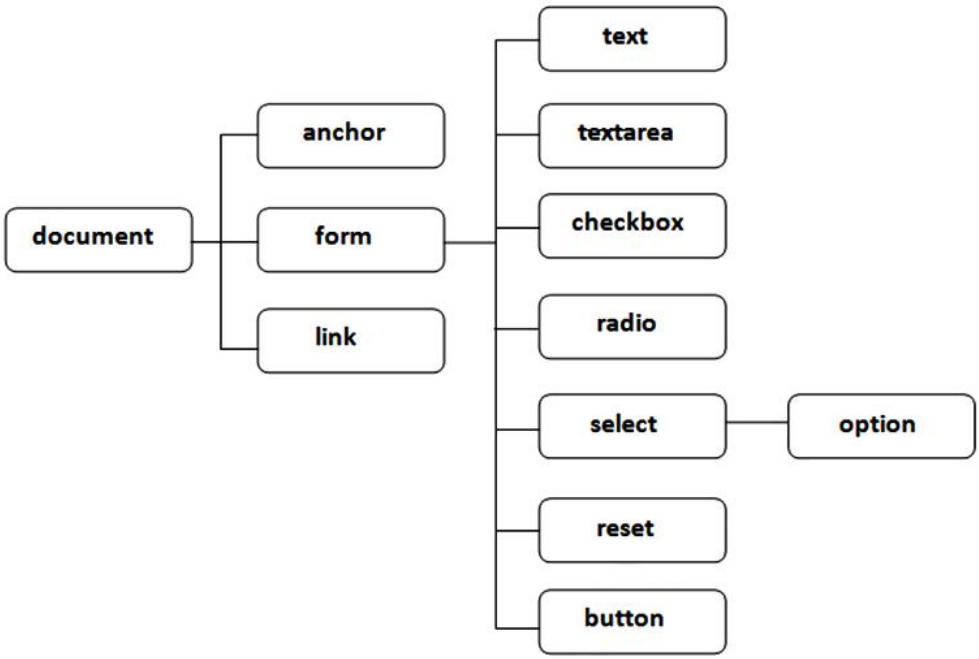
Is same as

document

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.

Accessing field value by document object

In this example, we are going to get the value of input text by user. Here, we are using document. form1.name.value to get the value of name field.

Here, document is the root element that represents the html document.

- form1 is the name of the form.
- name is the attribute name of the input text.
- value is the property, that returns the value of the input text.

Let's see the simple example of document object that prints name with welcome message.

```
<script type="text/javascript">
function printvalue(){
var name=document.form1.name.value;
```

```
alert("Welcome: "+name);
}
</script>
<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

Output of the above example

Enter Name:

Q24. Explain about window object.

Ans :

Window Object

The window object represents a window in browser. An object of window is created automatically by the browser. Window is the object of browser, it is not the object of javascript. The javascript objects are string, array, date etc.

Methods of Window Object

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

Example of alert() in javascript

It displays alert dialog box. It has message and ok button.

```
<script type="text/javascript">
function msg(){
alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

Example of confirm() in javascript

It displays the confirm dialog box. It has message with ok and cancel buttons.

```
<script type="text/javascript">
```

```
function msg(){
var v= confirm("Are u sure?");
if(v==true){
alert("ok");
}
else{
alert("cancel");
}
}
</script>
<input type="button" value="delete record" onclick="msg()"/>
```

Example of prompt() in javascript

It displays prompt dialog box for input. It has message and textfield.

```
<script type="text/javascript">
function msg(){
var v= prompt("Who are you?");
alert("I am "+v);
}
    </script>
<input type="button" value="click" onclick="msg()"/>
```

Example of open() in javascript

It displays the content in a new window.

```
<script type="text/javascript">
function msg(){
open("http://www.javatpoint.com");
}
</script>
<input type="button" value="javatpoint" onclick="msg()"/>
```

Example of setTimeout() in javascript

It performs its task after the given milliseconds.

```
<script type="text/javascript">
function msg(){
setTimeout(
function(){
alert("Welcome to Javatpoint after 2 seconds")
},2000);
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

3.3.7 Using Cookies

Q25. What are cookies? How to create a Cookie in JavaScript?

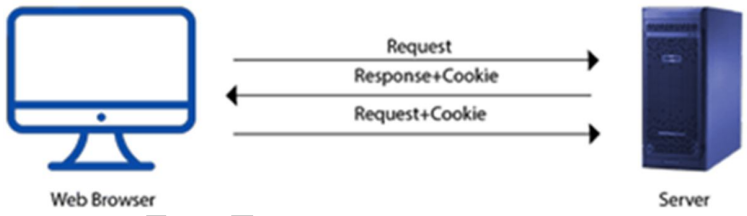
Ans :

A cookie is an amount of information that persists between a server-side and a client-side. A web browser stores this information at the time of browsing.

A cookie contains the information as a string generally in the form of a name-value pair separated by semi-colons. It maintains the state of a user and remembers the user's information among all the web pages.

How Cookies Works?

- When a user sends a request to the server, then each of that request is treated as a new request sent by the different user.
- So, to recognize the old user, we need to add the cookie with the response from the server.
- Browser at the client-side.
- Now, whenever a user sends a request to the server, the cookie is added with that request automatically. Due to the cookie, the server recognizes the users.



In JavaScript, we can create, read, update and delete a cookie by using **document.cookie** property.

The following syntax is used to create a cookie:

```
document.cookie="name=value";
```

JavaScript Cookie Example

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <input type="button" value="setCookie" onclick="setCookie()">
    <input type="button" value="getCookie" onclick="getCookie()">
<script>
function setCookie()
{
    document.cookie="username=Duke Martin";
```

```
    }
    function getCookie()
    {
        if(document.cookie.length!=0)
        {
        alert(document.cookie);
        }
        else
        {
        alert("Cookie not available");
        }
    }
    </script>
  </body>
</html>
```

Example 2

Here, we display the cookie's name-value pair separately.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <input type="button" value="setCookie"
onclick ="setCookie()">
    <input type="button" value= "getCookie"
onclick = "getCookie()">
    <script>
    function setCookie()
    {
        document.cookie="username=Duke
         Martin";
    }
    function getCookie()
    {
        if(document.cookie.length!=0)
        {
    var array=document. cookie.split ("=");
    alert("Name="+array[0]+" "+
    "Value="+array[1]);
    }
    else
    {
    alert("Cookie not available");
    }
    }
    </script>
  </body>
</html>
```

Short Question & Answers

1. What is array in javascript?

Ans :

Javascript Array is a global object which contains a list of elements. It is similar to other variables where they hold any type of data according to data type declaration but the difference is Array can hold more than one item at a time. Javascript allows a declaration of an array in many ways. Most important and common among those are 'Using array constructor' and 'Using literal notation'.

Syntax:

The array takes a list of items separated by a comma and enclosed in square brackets.

var <array_name> = [element0, element1, element2, element3, …..];

Elements inside an array can be of any data type, like string, numbers, Boolean, objects, etc which means array can consist of a string data type in its first position, number in the second, Boolean value in third and so on. Arrays in javascript are starting from index 0 and hence known as zero-based.

Here, we can see that index 0 and index 1 have string values whereas index 2 has numbers.

2. Event handling in java script.

Ans :

Events are actions performed by the user such as clicking a button or hovering the mouse over a link. There are several events performed on an HTML page. You can override all of these events and use a custom JavaScript function to display feedback to the user. We've discussed some basic event handling, but you can also register event handlers with the JavaScript engine to handle more complex actions. These events can perform a simple change such as switching colors in a paragraph or returning values to the user depending on the element clicked.

Registering Event Handlers

You can also register event handlers when you have more complex functions you want to hook to your HTML elements. The advantage of registering event handlers within the JavaScript code is that we can use the "this" statement. Instead of passing a value or element to the JavaScript function like we did in previous examples, we can leave out parameters and use "this" to manipulate properties for the element.

Registering events and hooking them to functions only takes one line of code. The following code registers and event to a button named "myButton."

myButton.onclick = ChangeColor;

Now, you define the function named "Change Color." Notice that you don't add the parenthesis to the button event handler registration. This is a part of the event handler registration process in JavaScript and other languages. If you add the parenthesis, the JavaScript engine things you are trying to run a function and assign a value to the onclick method.

3. Event Onload

Ans :

In JavaScript, this event can apply to launch a particular function when the page is fully displayed. It can also be used to verify the type and version of the visitor's browser. We can check what cookies a page uses by using the onload attribute.

In HTML, the onload attribute fires when an object has been loaded. The purpose of this attribute is to execute a script when the associated element loads.

In HTML, the **onload** attribute is generally used with the **<body>** element to execute a script once the content (including CSS files, images, scripts, etc.) of the webpage is completely loaded. It is not necessary to use it only with <body> tag, as it can be used with other HTML elements.

The difference between the document. onload and window.onload is: document. onload triggers before the loading of images and other external content. It is fired before the window.onload. While the window.onload triggers when the entire page loads, including CSS files, script files, images, etc.

Syntax

```
window.onload = fun()
```

4. Onfocus event in javascript.

Ans :

The onfocus event occurs when an element gets focus.

The onfocus event is most often used with <input>, <select>, and <a>.

This attribute is commonly used with elements such as <input>, <a>, <select>. This event attribute is sustained by every HTML element except the ones like <base>, <head>, <bdo>,
, <html>, <meta>, <iframe>, <param>, <style>, <script>, and <title> elements. This event handler executes when an element becomes focus on the screen of user. The elements focus can be when the user clicks on a text box. At that time, it is in focus by the user, since the person has clicked on it.

Syntax

Given below is the syntax of the onfocus event handler:

```
<element onfocus = "script">
```

Here, the attribute value is the script value that runs when the attribute onfocus is called.

5. Onsubmit event in javascript with example.

Ans :

The JavaScript onsubmit is one of the event handling function used for performing the operations in web based applications. It is used to validate the datas and whatever the client user is requested to the server it is to be validated and mainly used in the html form controls also it is used to set the html contents of the some input like hidden elements is to be validated before the request is submitting to the server side. If suppose the user is forget to submit the required fields in the form they can be cancelled the submission within the onsubmit event.

Syntax

Every event handling function must have their own syntax and rules in the JavaScript code.

```
<html>
<head>
<script>
function functionName()
{
```

```
— some javascript logic codes —
}
</script>
</head>
<body>
<form id="" method="" action="" onsubmit=" functionName()">
— some html input tags and elements —
</form>
</body>
</html>
```

The above code is the basic syntax for using onsubmit().

6. What is event bubbling in java script?

Ans :

Event Bubbling

While developing a webpage or a website via JavaScript, the concept of event bubbling is used where the event handlers are invoked when one element is nested on to the other element and are part of the same event. This technique or method is known as Event Bubbling. Thus, while performing event flow for a web page, event bubbling is used. We can understand event bubbling as a sequence of calling the event handlers when one element is nested in another element, and both the elements have registered listeners for the same event. So beginning from the deepest element to its parents covering all its ancestors on the way to top to bottom, calling is performed.

7. Onresize event.

Ans :

JavaScript onresize event

The onresize event in JavaScript generally occurs when the window has been resized. To get the size of the window, we can use the JavaScript's window.outer Width and window.outer Height events. We can also use the JavaScript's properties such as inner Width, innerHeight, clientWidth, ClientHeight, offsetWidth, offsetHeight to get the size of an element.

In HTML, we can use the onresize attribute and assign a JavaScript function to it. We can also use the JavaScript's addEventListener() method and pass a resize event to it for greater flexibility.

Syntax

Now, we see the syntax of using the **onresize** event in HTML and in javascript (without **addEventListener()** method or by using the **addEventListener()** method).

In HTML

<element onresize = "fun()"**>**

In JavaScript

object.onresize = function() { myScript };

In JavaScript by using the addEventListener() method

object.addEventListener("resize",my Script);

8. Javascript objects.

Ans :

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

9. Explain String object in Java script.

Ans :

The JavaScript string is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

1. By string literal

The string literal is created using double quotes. The syntax of creating string using string literal is given below:

var stringname="string value";

Let's see the simple example of creating string literal.

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

Output

This is string literal

2. By string object (using new keyword)

The syntax of creating string object using new keyword is given below:

var stringname=new String("string literal");

Here, **new keyword** is used to create instance of string.

Let's see the example of creating string in JavaScript by new keyword.

```
<script>
var stringname=new String("hello javascript string");
document.write(stringname);
</script>
```

Output

hello javascript string

10. Boolean and number object.

Ans :

JavaScript Boolean is an object that represents value in two states: true or false. You can create the JavaScript Boolean object by Boolean() constructor as given below.

Boolean b=new Boolean(value);

The default value of JavaScript Boolean object is *false*.

JavaScript Boolean Example

```
<script>
document.write(10<20);//true
document.write(10<5);//false
</script>
```

JavaScript Boolean Properties

Property	Description
Constructor	Returns the reference of Boolean function that created Boolean object.
Prototype	Enables you to add properties and methods in Boolean prototype.

JavaScript Boolean Methods

Method	Description
toSource()	returns the source of Boolean object as a string.
toString()	converts Boolean into String.
valueOf()	converts other type into Boolean.

JavaScript Number Object

The JavaScript number object enables you to represent a numeric value. It may be integer or floating-point. JavaScript number object follows IEEE standard to represent the floating-point numbers.

By the help of Number() constructor, you can create number object in JavaScript. For example:

```
var n=new Number(value);
```

If value can't be converted to number, it returns NaN(Not a Number) that can be checked by isNaN() method.

You can direct assign a number to a variable also. For example:

```
var x=102;//integer value
var y=102.7;//floating point value
var z=13e4;//exponent value, output: 130000
var n=new Number(16);//integer value by number object
```

Output

102 102.7 130000 16

11. Document Object Model.

Ans :

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.

As mentioned earlier, it is the object of window. So

window.document

Is same as

document

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."

12. window object.

Ans :

Window Object

The window object represents a window in browser. An object of window is created automatically by the browser. Window is the object of browser, it is not the object of javascript. The javascript objects are string, array, date etc.

Methods of Window Object

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

13. What are cookies?

Ans :

A cookie is an amount of information that persists between a server-side and a client-side. A web browser stores this information at the time of browsing.

A cookie contains the information as a string generally in the form of a name-value pair separated by semi-colons. It maintains the state of a user and remembers the user's information among all the web pages.

How Cookies Works?

- When a user sends a request to the server, then each of that request is treated as a new request sent by the different user.
- So, to recognize the old user, we need to add the cookie with the response from the server.
- Browser at the client-side.
- Now, whenever a user sends a request to the server, the cookie is added with that request automatically. Due to the cookie, the server recognizes the users.

Choose the Correct Answer

1. A collection of elements of the same data type which may either in order or not, is called _____. [b]

   (a) String (b) Array
   (c) Serialized Object (d) Object

2. Which of the following function of the String object returns the character in the string starting at the specified position via the specified number of characters? [c]

   (a) slice() (b) split()
   (c) substr() (d) search()

3. Which one of the given options can be considered as a code equivalent to the following code?

   var o =newObject(); [c]

   (a) var o= new Object; (b) var o;
   (c) var o = Object(); (d) Object o=new Object();

4. Which of the following is a server-side JavaScript object? [c]

   (a) Date (b) FileUpload
   (c) File (d) Function

5. Which of the following built-in method is used to remove the last element from an array and return that element? [b]

   (a) last() (b) pop()
   (c) get() (d) None of the above.

6. What are the different types of Pop up boxes available in JavaScript? [d]

   (a) Alert (b) Prompt
   (c) Confirm (d) All of the above

7. Which of the following syntax can be used to write “Hello World” in an alert box? [c]

   (a) alertBox(“Hello World”); (b) msgBox(“Hello World”);
   (c) alert(“Hello World”); (d) msg(“Hello World”);

8. Which of the following function of the Array object is used to add one or more elements to the front of an array and returns the new length of the array? [b]

   (a) splice() (b) unshift()
   (c) sort() (d) toString()

9. Which of the following is the correct syntax to display “Letsfindcourse” in an alert box using JavaScript? [d]

   (a) alert-box(“Letsfindcourse”); (b) confirm(“Letsfindcourse”);
   (c) msgbox(“Letsfindcourse”); (d) alert(“Letsfindcourse”);

10. Out of the following functions of the string object, which one would return the character in any string via the specified number of characters starting at a specified position? [b]

    (a) search() (b) substr()
    (c) split() (d) slice()

Fill in the blanks

1. To know about an object, whether the object is a prototype (or a part of a prototype chain) of another object, the user can use _____.
2. In the following line of code, what we will call the "datatype" written in brackets is _____.

   article[datatype]=assignment_value;
3. The linkage of a set of prototype objects is known as _____.
4. _____ are the events that have default actions that can be canceled by event handlers?
5. The events that represent occurrences related to the browser window are _____.
6. You can refresh the webpage in JavaScript by using _____.
7. When document loads in javascript then _____ method is used to access the screen when a page is loading.
8. _____ Event is fired while scrolling a scrollable document element.
9. OnClick event is used to handle _____.
10. An _____ is invoked by the browser when that specified event occurs

ANSWERS

1. isPrototypeOf() method
2. An String
3. prototype chain
4. Submit and reset events
5. Window
6. location.reload
7.
8. scroll
9. Handle click events
10. event handler

    window.onload

UNIT IV

XML - Introduction, XML Basics, Structuring Data, XML Namespaces, Document Type Definitions (DTDs), W3C XML Schema Documents, XML Vocabularies, Extensible Style sheet Language and XSL Transformations, Document Object Model (DOM).

Ajax-Enabled Rich Internet Applications: introduction, history of Ajax, traditional web applications Vs Ajax Applications, RIAs with Ajax, Ajax example using XML HttpRequest object, XML and DOM, creating full scale Ajax-enabled application, Dojo Toolkit.

4.1 XML

4.1.1 Introduction

Q1. What is XML? Write about it.

Ans : **(Imp.)**

- XML (extensible Markup Language) is a mark up language.
- XML is designed to store and transport data.
- Xml was released in late 90's. it was created to provide an easy to use and store self describing data.
- XML became a W3C Recommendation on February 10, 1998.
- XML is not a replacement for HTML.
- XML is designed to be self-descriptive.
- XML is designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is platform independent and language independent.

Why xml

Platform Independent and Language Independent: The main benefit of xml is that you can use it to take data from a program like Microsoft SQL, convert it into XML then share that XML with other programs and platforms. You can communicate between two platforms which are generally very difficult.

The main thing which makes XML truly powerful is its international acceptance. Many corporation use XML interfaces for databases, programming, office application mobile phones and more. It is due to its platform independent feature.

4.1.2 XML Basics

Q2. Write the features of XML.

Ans : **(Imp.)**

XML is widely used in the era of web development. It is also used to simplify data storage and data sharing.

The main features or advantages of XML are given below.

1. XML separates data from HTML

If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes.

With XML, data can be stored in separate XML files. This way you can focus on using HTML/CSS for display and layout, and be sure that changes in the underlying data will not require any changes to the HTM

With a few lines of JavaScript code, you can read an external XML file and update the data content of your web page.

2. XML simplifies data sharing

In the real world, computer systems and databases contain data in incompatible formats.

XML data is stored in plain text format. This provides a software- and hardware independent way of storing data.

This makes it much easier to create data that can be shared by different applications.

3. XML simplifies data transport

One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet.

Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

4. XML simplifies Platform change

Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost.

XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

5. XML increases data availability

Different applications can access your data, not only in HTML pages, but also from XML data sources.

With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.

6. XML can be used to create new internet languages

A lot of new Internet languages are created with XML.

Here are some examples:

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

Q3. Explain the syntax rules of XML.

Ans :

XML Syntax Rules

The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.

XML Documents Must Have a Root Element

XML documents must contain one **root** element that is the **parent** of all other elements:

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

In this example **<note>** is the root element:

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</
    body>
</note>
```

The XML Prolog

- This line is called the XML **prolog**:
- <?xml version="1.0" encoding="UTF-8"?>
- The XML prolog is optional. If it exists, it must come first in the document.
- XML documents can contain international characters, like Norwegian **øæå** or French **êèé.**
- To avoid errors, you should specify the encoding used, or save your XML files as UTF-8.
- UTF-8 is the default character encoding for XML documents.

All XML Elements Must Have a Closing Tag

- In XML, it is illegal to omit the closing tag. All elements **must** have a closing tag:
  - This is a paragraph.
  -
- XML Tags are Case Sensitive
  - XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.
  - Opening and closing tags must be written with the same case:
- <message>This is correct</message>
- "Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing
  - XML Elements Must be Properly Nested
- In HTML, you might see improperly nested elements:
  - This text is bold and italic
- In XML, all elements **must** be properly nested within each other:
  - This text is bold and italic

In the example above, "Properly nested" simply means that since the element is opened inside the element, it must be closed inside the element.

- XML Attribute Values Must Always be Quoted
- XML elements can have attributes in name/value pairs just like in HTML.
  - In XML, the attribute values must always be quoted:

```
<note date="12/11/2007">
    <to>Tove</to>
    <from>Jani</from>
</note>
```

Entity References

Some characters have a special meaning in XML.

If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

<message>salary < 1000</message>

To avoid this error, replace the "<" character with an **entity reference**:

<message>salary < 1000</message>

There are 5 pre-defined entity references in XML:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Comments in XML

The syntax for writing comments in XML is similar to that of HTML:

<!— This is a comment —>

Two dashes in the middle of a comment are not allowed:

<!— This is an invalid — comment —>

White-space is Preserved in XML

- XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces to one single white-space):
- XML Stores New Line as LF
- XML documents that conform to the syntax rules above are said to be "Well Formed" XML documents.

What is an XML Element?

An XML element is everything from (including) the element's start tag to (including) the element's end tag.

<price>29.99</price>

An element can contain

- text
- attributes
- other elements
- or a mix of the above

```
<bookstore>
    <book category="children">
        <title>Harry Potter</title>
        <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
    </book>
    <book category="web">
        <title>Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```

In the example above

<title>, <author>, <year>, and <price> have text content because they contain text (like 29.99).

<bookstore> and <book> have element contents, because they contain elements.

<book> has an attribute (category=" children").

Empty XML Elements

An element with no content is said to be empty.

In XML, you can indicate an empty element like this:

<element></element>

You can also use a so called self-closing tag:

<element />

XML Naming Rules

XML elements must follow these naming rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

Any name can be used, no words are reserved (except xml).

Best Naming Practices

Create descriptive names, like this: <person>, <firstname>, <lastname>.

Create short and simple names, like this: <book_title> not like this: <the_title_of_ the_ book>.

Avoid "-". If you name something "first-name", some software may think you want to subtract "name" from "first".

Avoid ".". If you name something "first.name", some software may think that "name" is a property of the object "first".

Avoid ":". Colons are reserved for namespaces (more later).

Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software doesn't support them!

Naming Conventions

Some commonly used naming conventions for XML elements:

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Snake case	<first_name>	Underscore separates words (commonly used in SQL databases)
Pascal case	<FirstName>	Uppercase first letter in each word (commonly used by C programmers)
Camel case	<firstName>	Uppercase first letter in each word except the first (commonly used in JavaScript)

- XML documents often have a corresponding database. A common practice is to use the naming rules of the database for the XML elements.
- XML Elements are Extensible
- XML elements can be extended to carry more information.

Look at the following XML example:

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <body>Don't forget me this weekend!</body>
</note>
```

Let's imagine that we created an application that extracted the <to>, <from>, and <body> elements from the XML document to produce this output:

MESSAGE

To: Tove

From: Jani

Don't forget me this weekend!

Imagine that the author of the XML document added some extra information to it:

```
<note>
    <date>2008-01-10</date>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

XML Attributes

- XML elements can have attributes. By the use of attributes we can add the information about the element.
- XML attributes enhance the properties of the elements.

Let us take an example of a book publisher. Here, book is the element and publisher is the attribute.

<book publisher= "Tata McGraw Hill" **></book>**

Or

<book publisher= 'Tata McGraw Hill'**></book>**

Metadata should be stored as attribute and data should be stored as element

```
<book>
    <book category="computer">
    <author> A & B </author>
</book>
```

Data can be stored in attributes or in child elements. But there are some limitations in using attributes, over child elements.

XML Comments

XML comments are just like HTML comments. We know that the comments are used to make codes more understandable other developers.

XML Comments add notes or lines for understanding the purpose of an XML code. Although XML is known as self-describing data but sometimes XML comments are necessary.

Syntax

An XML comment should be written as:

```
<!— Write your comment—>
```

XML Comments Example

Let's take an example to show the use of comment in an XML example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!–Students marks are uploaded by months>
<students>
    <student>
        <name>Ratan</name>
        <marks>70</marks>
    </student>
    <student>
        <name>Aryan</name>
        <marks>60</marks>
    </student>
</students>
```

Rules for adding XML comments

- Don't use a comment before an XML declaration.
- You can use a comment anywhere in XML document except within attribute value.
- Don't nest a comment inside the other comment.

4.1.3 Structuring Data

Q4. Explain the structure of XML data.

Ans :

In XML document has a self descriptive structure. It forms a tree structure which is referred as an XML tree. The tree structure makes easy to describe an XML document.

A tree structure contains root element (as parent), child element and so on. It is very easy to traverse all succeeding branches and sub-branches and leaf nodes starting from the root.

1. An XML document contains exactly one root element: the start tag of the XML document, and it contains all other elements.

```
<root>
    <section>
        <sub-section></sub-section>
        <sub-section></sub-section>
    </section>
    <section>
        <sub-section></sub-section>
        <sub-section></sub-section>
    </section>
<root>
```

2. XML documents may begin with a prolog that appears before the root element. It has the metadata about the XML document, such as character encoding, document structure, and style sheets. For example,

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

3. A tag in XML is a case-sensitive markup construct that begins with < and ends with >. A tag can be:
   - A start-tag, such as <name>
   - An end-tag, such as </name>
   - An empty-element tag, such as <name>

4. An element in XML is formed by characters between the start-tag and the end-tag. For example, <name>John Snow</name>. It can also consist only of an empty-element tag. For example, <name/>.

5. XML elements can have attributes which exists within a start-tag or empty-element tag.

   An attribute consists of a name–value pair. For example,

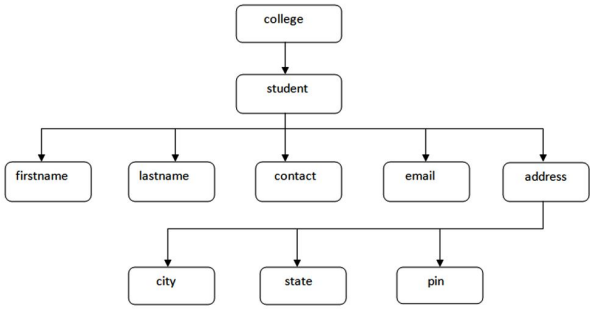
   1

6. Here the names of the attributes are *src* and *alt*, and their values are *screenshot.png* and *screenshot*, respectively.

Example of an XML document

```
<?xml version="1.0"?>
<college>
    <student>
        <firstname>Tamanna</firstname>
        <lastname>Bhatia</lastname>
        <contact>09990449935</contact>
        <email>tammanabhatia@abc.com</email>
        <address>
            <city>Ghaziabad</city>
            <state>Uttar Pradesh</state>
            <pin>201007</pin>
        </address>
    </student>
</college>
```

Let's see the tree-structure representation of the above example.



In the above example, first line is the XML declaration. It defines the XML version 1.0. Next line shows the root element (college) of the document. Inside that there is one more element (student). Student element contains five branches named <firstname>, <lastname>, <contact>, <Email> and <address>.

4.1.4 XML Namespaces

Q5. Write about XML namespaces.

Ans :

XML Namespace is used to avoid element name conflict in XML document.

XML Namespace Declaration

An XML namespace is declared using the reserved XML attribute. This attribute name must be started with "xmlns".

syntax:<element xmlns:name = "URL">

Here, namespace starts with keyword "xmlns". The word name is a namespace prefix. The URL is a namespace identifier.

Let's see the example of XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<cont:contact xmlns:cont="http://sssit.org/contact-us">
<cont:name>Vimal Jaiswal</cont:name>
<cont:company>SSSIT.org</cont:company>
<cont:phone>(0120) 425-6464</cont:phone>
</cont:contact>
```

Namespace Prefix: cont

Namespace Identifier: http://sssit.org/contact-us

It specifies that the element name and attribute names with cont prefix belongs to http://sssit.org/contact-us name space.

In XML, elements name are defined by the developer so there is a chance to conflict in name of the elements. To avoid these types of confliction we use XML Namespaces. We can say that XML Namespaces provide a method to avoid element name conflict.

Generally these conflict occurs when we try to mix XML documents from different XML application.

Let's take an example with two tables:

Table1

```
<table>
    <tr>
        <td>Aries</td>
        <td>Bingo</td>
    </tr>
</table>
```

Table2: This table carries information about a computer table.

```
<table>
    <name>Computer table</name>
    <width>80</width>
    <length>120</length>
</table>
```

If you add these both XML fragments together, there would be a name conflict because both have <table< element. Although they have different name and meaning.

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>
```

2. By Using xmlns Attribute

You can use xmlns attribute to define namespace with the following syntax:

```
<element xmlns:name = "URL">
<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.

```
<root xmlns:h="http://www.abc.com/TR/
html4/"
xmlns:f="http://www.xyz.com/furniture">
<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>
</root>
```

The Default Namespace

The default namespace is used in the XML document to save you from using prefixes in all the child elements.

The only difference between default namespace and a simple namespace is that: There is no need to use a prefix in default namespace.

You can also use multiple namespaces within the same document just define a namespace against a child node.

Example of Default Namespace:

```
<tutorials xmlns="http://www.java.com/
java-tutorial">
<tutorial>
<title>Java-tutorial</title>
<author>Sonoo Jaiswal</author>
</tutorial>
...
</tutorials>
```

You can see that prefix is not used in this example, so it is a default namespace.

4.1.5 Document Type Definitions (DTDS)

Q6. Write about XML validation.

Ans :

XML Validation

A well formed XML document can be validated against DTD or Schema.

A well-formed XML document is an XML document with correct syntax. It is very necessary to know about valid XML document before knowing XML validation.

Valid XML document

- It must be well formed (satisfy all the basic syntax condition)
- It should be behave according to predefined DTD or XML schema

Rules for well formed XML

- It must begin with the XML declaration.
- It must have one unique root element.
- All start tags of XML documents must match end tags.
- XML tags are case sensitive.
- All elements must be closed.
- All elements must be properly nested.
- All attributes values must be quoted.
- XML entities must be used for special characters.

XML DTD

A DTD defines the legal elements of an XML document

- In simple words we can say that a DTD defines the document structure with a list of legal elements and attributes.
- XML schema is a XML based alternative to DTD.
- Actually DTD and XML schema both are used to form a well formed XML document.
- We should avoid errors in XML documents because they will stop the XML programs.

XML schema

- It is defined as an XML language
- Uses namespaces to allow for reuses of existing definitions
- It supports a large number of built in data types and definition of derived data types.

Q7. Explain XML DTD with an example.

Ans :

Document Type Definition (DTD) defines the schema of an XML document which includes elements, attributes in it. If the XML documents are conformed to the DTD format then it is valid and it is used in business-to-business applications where XML documents are exchanged in which they are defined using extended Backus-Naur form. Multiple documents and different applications share DTDS also defines the order of elements. DTD are defined in the Document with the declaration <!DOCTYPE > and each XML document holds one DTD.

How DTD Works in XML?

DTD is also the schema language preferred in mark up language. The general Syntax is given below:

<!DOCTYPE root element name

[

declaration1

declaration2

<!ELEMENT><ATTLIST><!ENTITY><!NOTATION>

]>

In the above syntax, the DTD name is the root element name and followed by options which say about the schemas and types. The keyword! DOCTYPE should be uppercase. Let's see Element declarations.

The working of DTD is performed by the following steps:

- First, create a DTD file for the respective XML Document.
- Next outline the structure of the document.
- Initiate with the root node which is the same as DOCTYPE. You can create DTD either internal or external references.

- Include all the elements, attributes, entities for the file. The parser eliminates empty elements.

Element Declarations

The element specifications with the sequence of its elements are stated as

<!ELEMENT name (seq of elements)>

Ex: <!ELEMENT ship (crew name, location, date)> // this statement is often termed as generic identifier.

Also, the element specifies the number of occurrences of the child elements using (+, *,?).

- **+** sign indicates two or more frequency
- ***indicates the element can appear more number of times.
- **?** Indicates elements can occur only once.

The following declaration is

<!ELEMENT pizza (onion? Cheese* ((veg |noveg) + |topping))>

The above statement implies that the pizza element can have one onion elements followed by one or more cheese and so on.

The respective XML file could be

<pizza>

<onion> fried </onion>

<cheese> thin </cheese>

<cheese> thick </cheese>

<topping>oregano </topping>

</pizza>

1. Attribute Declarations

The attributes for a given element is designed by the following rule:

<!ATTLIST element name attribute names 1 attribute type restriction/Default >

Example

Here attribute is specified using the keyword ATTLIST, the element name is included for the respective attributes unless they are optional. The attribute types include PCDATA, tokens, entity, notation. Last is restriction/ default they are placed based on the occurrences of the values. The attribute default includes #IMPLIED, #REQUIRED, #FIXED. The implied specifies the attribute value doesn't appear and required implies the attribute value is present and fixed denotes a constant value.

2. Defining Entities

Entity is used to specify special characters. The entity declaration is

<!ENTITY entity_name SYSTEM "URL path">

Example

<!ENTITY SYSTEM "http://www.ckjd.com/pot.dtd">

As DTD is model of the XML document it talks about the elements, attributes being used which are essential and optional as they are easy to validate the document and there are two types of DTDs namely,

- Internal DTD
- External DTD

3. Internal DTD

This type of DTD is declared inside the XML Document. It is declared as.

<!DOCTYPE root name [DTD related specification]>

Example: <!DOCTYPE healthcare [

<!ELEMENT healthcare (#PCDATA)>]>

The content inside the square brackets is considered to be the internal subset. And the keyword! ELEMENT is element declarations, PCDATA is the parsed character data which are parsed by the XML parsers.

4. External DTD

This type of DTD is declared outside the XML file with a separate file. External DTD is used in multiple XML documents, the updation done in this file affects all the XML document which is quite easy while changing the input file. In external DTD the 'standalone' keyword

is set to "no". The external content is specified using a keyword 'PUBLIC' and 'SYSTEM'. The public keyword is used outside the XML document followed by a URL (specifies the path).

Note: Multiple DTDs are allowed in which both external and internal DTDs are combined.

<!DOCTYPE root-name SYSTEM " XML file-name">

There are two types of External DTD: Private and public.

Examples to Implement DTD in XML

Following are the examples of dtd in xml are given below:

Example #1 – External DTD

Here the DTD file is created external and saved as stck.dtd and the corresponding element name is declared in the separate XML file.

stck.dtd

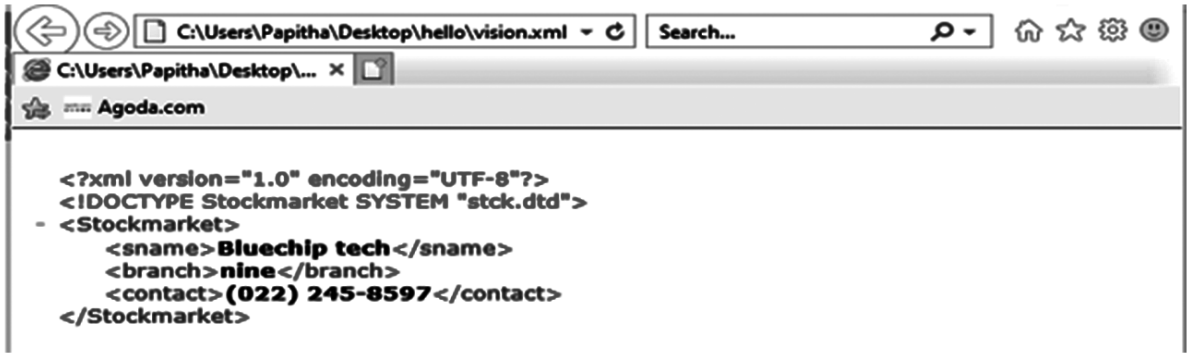
```
<?xml version="1.0"?>
<!ELEMENT Stockmarket (sname,branch,contact)>
<!ELEMENT sname (#PCDATA)>
<!ELEMENT branch (#PCDATA)>
<!ELEMENT contact (#PCDATA)>
```

XML file

vision.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
<!DOCTYPE Stockmarket SYSTEM "stck.dtd">
<Stockmarket>
<sname>Bluechip tech</sname>
<branch>nine</branch>
<contact>(022) 245-8597</contact>
</Stockmarket>
```

Output



Example #2 – Using Entities

actor.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE Actor [
<!ELEMENT Actor ((aname)+, hollywood)>
<!ELEMENT aname (fname, lname)>
<!ELEMENT fname (#PCDATA)>
<!ELEMENT lname (#PCDATA)>
<!ELEMENT hollywood (#PCDATA)>
<!ENTITY UofT SYSTEM "fil.xml">
]>
<Actor>
<aname>
<fname>Markdev</fname>
<lname>carylon</lname>
</aname>
&UofT;
</Actor>
```

fil.xml

```
<holywood>
Mark of the film industry
</holywood>
```

Output

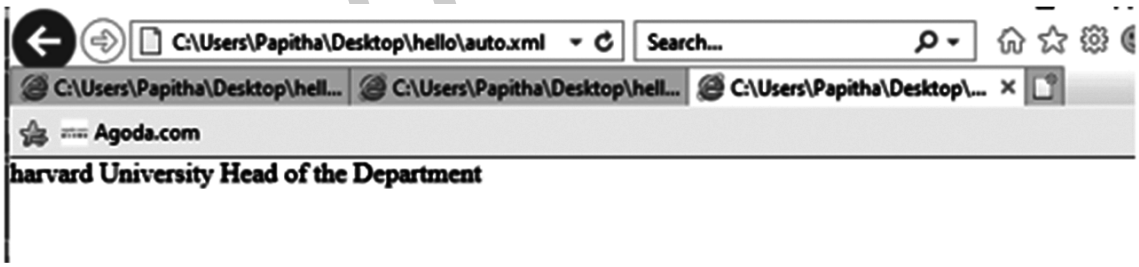
Example #3

Following is an XML file with DTD declared inside the XML file-Internal DTD which is embedded inside the keyword DOCTYPE. In the below example the element node university has three fields and those are declared of the type PCDATA.

auto.xml

```
<?XML version= “1.0" standalone= “yes”?>
<!DOCTYPE University [
<!ELEMENT University (collegename, type, email)>
<!ATTLIST University
id CDATA #REQUIRED>
<!ELEMENT collegename (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ATTLIST University
department CDATA #IMPLIED>
<!ELEMENT email (#PCDATA)>
]>
<University id=”02">
<collegename>Harvard University</collegename>
<type department=”Science and Humani-ties” >Head of the Department</type>
<email>yhkhi12@myhotmail.com</email>
</University >
```

Output



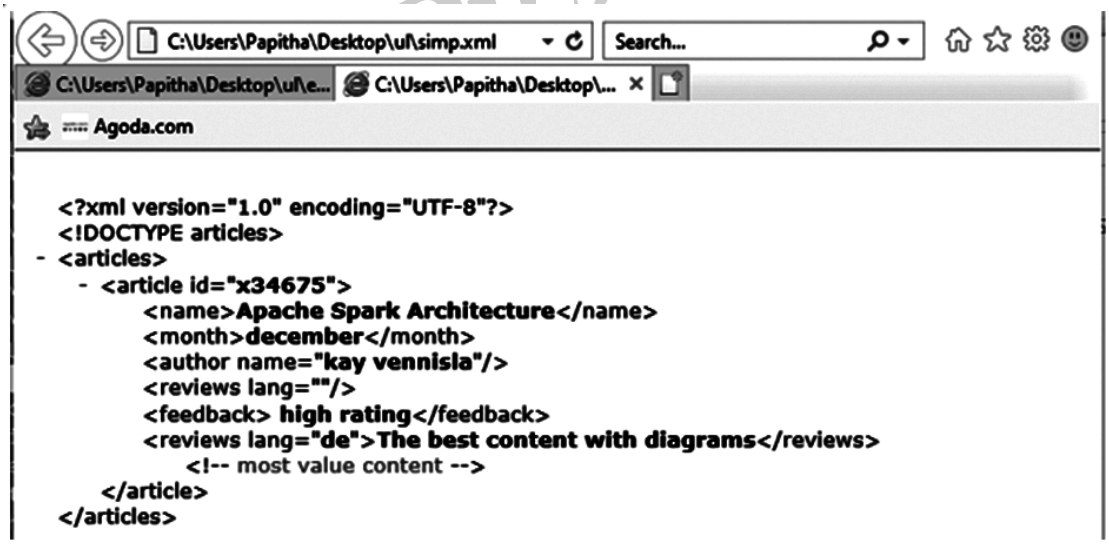
Example #4 – Internal DTD

simp.xml

```
<?xml version=”1.0" encoding=”UTF-8"?>
<!DOCTYPE articles [
<!ELEMENT articles (Movie+) >
<!ELEMENT article ( name, month, author, (reviews | feedback)+)>
<!ATTLIST article id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT month (#PCDATA) >
<!ELEMENT author (#PCDATA)>
```

```
<!ATTLIST author name CDATA #IMPLIED>
<!ELEMENT reviews (#PCDATA)>
<!ELEMENT feedback (#PCDATA)>
<!ATTLIST reviews lang CDATA #IMPLIED>
]>
<articles>
<article id="x34675">
<name>Apache Spark Architecture</name>
<month>december</month>
<author name="kay vennisla"/>
<reviews lang=""/>
<feedback > high rating</feedback>
<reviews lang="de">The best content with diagrams</reviews>
<!— most value content —>
</article>
</articles>
```

Output:



4.1.6 W3C XML Schema Documents

Q8. What is XML schema? Explain.

Ans :

XML schema is a language which is used for expressing constraint about XML documents. There are so many schema languages which are used now a days for example Relax- NG and XSD (XML schema definition).

An XML schema is used to define the structure of an XML document. It is like DTD but provides more control on XML structure.

Checking Validation

An XML document is called "well-formed" if it contains the correct syntax. A well-formed and valid XML document is one which have been validated against Schema.

XML Schema Example

Let's create a schema file *employee.xsd*

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.javatpoint.com"
xmlns="http://www.javatpoint.com"
elementFormDefault="qualified">
<xs:element name="employee">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="firstname" type="xs:string"/>
            <xs:element name="lastname" type="xs:string"/>
            <xs:element name="email" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Let's see the xml file using XML schema or XSD file.

employee.xml

```
<?xml version="1.0"?>
<employee
xmlns="http://www.javatpoint.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.javatpoint.com employee.xsd">
<firstname>vimal</firstname>
<lastname>jaiswal</lastname>
<email>vimal@javatpoint.com</email>
</employee>
```

Description of XML Schema

- **<xs:element name= "employee">** : It defines the element name employee.
- **<xs:complexType>** : It defines that the element 'employee' is complex type.
- **<xs:sequence>** : It defines that the complex type is a sequence of elements.
- **<xs:element name= "firstname" type="xs:string"/>** : It defines that the element 'firstname' is of string/text type.
- **<xs:element name= "lastname" type="xs:string"/>** : It defines that the element 'lastname' is of string/text type.
- **<xs:element name= "email" type="xs:string"/>** : It defines that the element 'email' is of string/text type.

XML Schema Data types

There are two types of data types in XML schema.

1. **simpleType :** The simpleType allows you to have text-based elements. It contains less attributes, child elements, and cannot be left empty.
2. **complexType :** The complexType allows you to hold multiple attributes and elements. It can contain additional sub elements and can be left empty.

4.1.7 XML Vocabularies

Q9. Write about XML vocabularies.

Ans :

XML vocabularies are the elements used in particular applications or data formats - the definitions of the meanings of those formats. For example, in CDF, element names such as <SCHEDULE>, <CHANNEL>, and <ITEM> make up the vocabulary for describing collections of pages, when these pages should be downloaded, etc.

Vocabularies, along with the structural relationships between the elements, are defined in XML DTDs or XML schemas.

XML elements can be defined as building blocks of an XML. Elements can behave as containers to hold text, elements, attributes, media objects or all of these.

Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

Syntax

```
<element-nameattribute1attribute2>
....content
</element-name>
```

where,

- **element-name** is the name of the element. The *name* its case in the start and end tags must match.
- **attribute1, attribute2** are attributes of the element separated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as "name = "value" *name* is followed by an = sign and a string *value* inside double(" ") or single(' ') quotes.

Empty Element

An empty element (element with no content) has following syntax "

<nameattribute1attribute2.../>

Following is an example of an XML document using various XML element "

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

XML Elements Rules

Following rules are required to be followed for XML elements

- An element *name* can contain any alphanumeric characters. The only punctuation mark allowed in names are the hyphen (-), under-score (_) and period (.).
- Names are case sensitive. For example, Address, address, and ADDRESS are different names.
- Start and end tags of an element must be identical.
- An element, which is a container, can contain text or elements as seen in the above example.

4.1.8 Extensible Style Sheet Language and XSL Transformations

Q10. What is XSL? Explain about it.

Ans : **(Imp.)**

In HTML documents, tags are predefined but in XML documents, tags are not predefined. World Wide Web Consortium (W3C) developed XSL to understand and style an XML document, which can act as XML based Stylesheet Language.

An XSL document specifies how a browser should render an XML document.

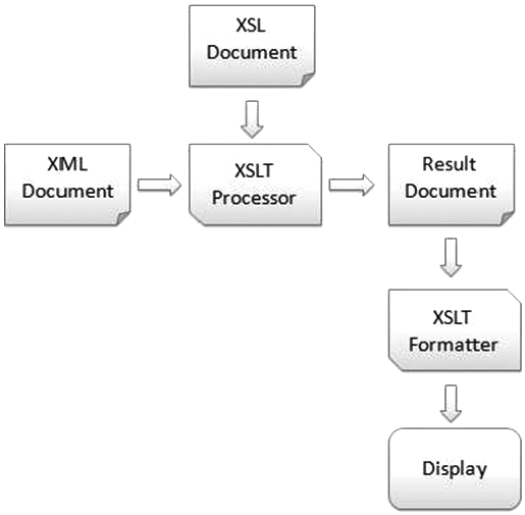
Main parts of XSL Document

- **XSLT:** It is a language for transforming XML documents into various other types of documents.
- **XPath:** It is a language for navigating in XML documents.
- **XQuery:** It is a language for querying XML documents.
- **XSL-FO:** It is a language for formatting XML documents.

How XSLT Works

The XSLT stylesheet is written in XML format. It is used to define the transformation rules to be applied on the target XML document. The 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. At the end it is used by XSLT formatter to generate the actual output and displayed on the end-user.

Image representation



Advantage of XSLT

- XSLT provides an easy way to merge XML data into presentation because it applies user defined transformations to an XML document and the output can be HTML, XML, or any other structured document.
- XSLT provides Xpath to locate elements/ attribute within an XML document. So it is more convenient way to traverse an XML document rather than a traditional way, by using scripting language.
- XSLT is template based. So it is more resilient to changes in documents than low level DOM and SAX.
- By using XML and XSLT, the application UI script will look clean and will be easier to maintain.
- XSLT templates are based on XPath pattern which is very powerful in terms of performance to process the XML document.
- XSLT can be used as a validation language as it uses tree-pattern-matching approach.
- You can change the output simply modifying the transformations in XSL files.

XSLT Syntax

Let's take an example to take a sample XML file and transform it into a well formatted HTML document.

See this example

Create an XML file named employee.xml, having the following code:

Employee.xml

```
<?xml version = "1.0"?>
<class>
   <employee id = "001">
      <firstname>Aryan</firstname>
      <lastname>Gupta</lastname>
      <nickname>Raju</nickname>
      <salary>30000</salary>
   </employee>
   <employee id = "024">
      <firstname>Sara</firstname>
      <lastname>Khan</lastname>
      <nickname>Zoya</nickname>
      <salary>25000</salary>
   </employee>
   <employee id = "056">
      <firstname>Peter</firstname>
      <lastname>Symon</lastname>
      <nickname>John</nickname>
      <salary>10000</salary>
   </employee>
</class>
```

Define an XSLT stylesheet document for the above XML document. You should follow the criteria give below:

- Page should have a title employee.
- Page should have a table of employee's details.
- Columns should have following headers: id, First Name, Last Name, Nick Name, Salary
- Table must contain details of the employees accordingly.

Step1: Create XSLT document

Create the XSLT document which satisfies the above requirements. Name it as employee.xsl and save it in the same location of employee.xml.

Employee.xsl

```
<?xml version = "1.0" encoding = "UTF-8"?>
    <!— xsl stylesheet declaration with xsl namespace:
    Namespace tells the xlst processor about which
    element is to be processed and which is used for output purpose only —>
    <xsl:stylesheet version = "1.0"
    xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
    <!— xsl template declaration:
    template tells the xlst processor about the section of xml
    document which is to be formatted. It takes an XPath expression.
    In our case, it is matching document root element and will
    tell processor to process the entire document with this template.—>
    <xsl:template match = "/">
    <!— HTML tags
    Used for formatting purpose. Processor will skip them and browser
    will simply render them. —>
<html>
    <body>
    <h2>Employee</h2>
    <table border = "1">
        <tr bgcolor = "#9acd32">
            <th>ID</th>
            <th>First Name</th>
        <th>Last Name</th>
        <th>Nick Name</th>
        <th>Salary</th>
    </tr>
    <!— for-each processing instruction
    Looks for each element matching the XPath expression —>
    <xsl:for-each select="class/employee">
        <tr>
        <td>
```

```
                    <!— value-of processing instruction
                    process the value of the element matching the XPath expression —>
                    <xsl:value-of select = "@id"/>
                            </td>
                                    <td><xsl:value-of select = "firstname"/></td>
                                    <td><xsl:value-of select = "lastname"/></td>
                                    <td><xsl:value-of select = "nickname"/></td>
                                    <td><xsl:value-of select = "salary"/></td>
                            </tr>
                        </xsl:for-each>
                    </table>
                </body>
            </html>
        </xsl:template>
</xsl:stylesheet>
```

Step2: List the XSLT document to the XML document

Update employee.xml document with the following xml-stylesheet tag. Set href value to employee.xsl

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "employee.xsl"?>
<class>
...
</class>
```

Updated "employee.xml"

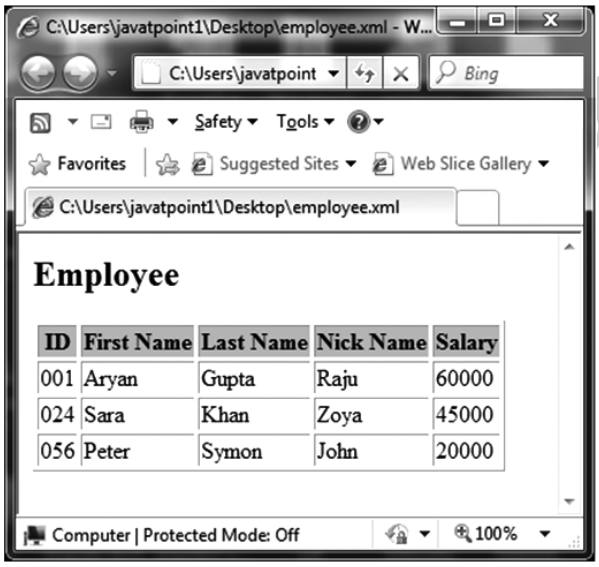
```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "employee.xsl"?>
<class>
    <employee id = "001">
        <firstname>Aryan</firstname>
        <lastname>Gupta</lastname>
        <nickname>Raju</nickname>
        <salary>30000</salary>
    </employee>
    <employee id = "024">
    <firstname>Sara</firstname>
```

```
<lastname>Khan</lastname>
<nickname>Zoya</nickname>
<salary>25000</salary>
</employee>
<employee id = ”056">
<firstname>Peter</firstname>
<lastname>Symon</lastname>
<nickname>John</nickname>
<salary>10000</salary>
</employee>
</class>
```

Step 3: View the XML document in Internet Explorer

The output will look like this:

Output:



XSLT <xsl:value-of> Element

The XSLT <xsl:value-of> element is used to extract the value of selected node. It puts the value of selected node as per XPath expression, as text.

<xsl:value-of

select = Expression

disable-output-escaping = “yes”| “no”**>**

</xsl:value-of>

Parameter explanation

1. Select It specifies the XPpath expression to be evaluated in current context.
2. Disable-outputescaping Default-”no”. If “yes”, output text will not escape XML characters from text.

XSLT <xsl:value-of> Element Example

Let’s take an example to see the usage of XSLT <xsl:value-of> element with its attribute “id” and its child <firstname>, <lastname>, <nickname>, and <salary>.

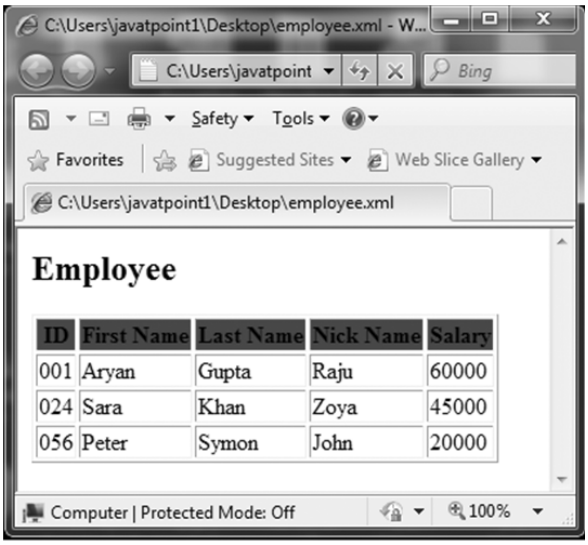
Employee.xml

```
<?xml version = “1.0"?>
<?xml-stylesheet type = ”text/xsl” href = “employee.xsl”?>
<class>
    <employee id = "001">
        <firstname>Aryan</firstname>
        <lastname>Gupta</lastname>
        <nickname>Raju</nickname>
        <salary>30000</salary>
    </employee>
    <employee id = "024">
        <firstname>Sara</firstname>
        <lastname>Khan</lastname>
        <nickname>Zoya</nickname>
        <salary>25000</salary>
    </employee>
    <employee id = "056">
        <firstname>Peter</firstname>
        <lastname>Symon</lastname>
        <nickname>John</nickname>
        <salary>10000</salary>
    </employee>
</class>
```

Employee.xsl

```
<?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>Employee</h2>
        <table border = "1">
                <tr bgcolor = "purple">
                    <th>ID</th>
                    <th>First Name</th>
                    <th>Last Name</th>
                    <th>Nick Name</th>
                    <th>Salary</th>
        </tr>
        <!— for-each processing instruction
        Looks for each element matching the XPath expression —>
        <xsl:for-each select="class/employee">
            <tr>
                <td>
                <!— value-of processing instruction
                process the value of the element matching the XPath expression—>
                <xsl:value-of select = "@id"/>
                </td>
                <td><xsl:value-of select = "firstname"/></td>
                <td><xsl:value-of select = "lastname"/></td>
                <td><xsl:value-of select = "nickname"/></td>
                <td><xsl:value-of select = "salary"/></td>
            </tr>
        </xsl:for-each>
    </table>
    </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

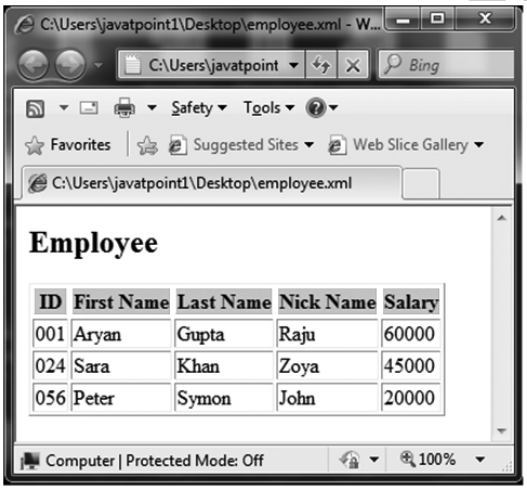
Output




```
<?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>Employee</h2>
            <table border = "1">
            <tr bgcolor = "purple">
                <th>ID</th>
                <th>First Name</th>
                <th>Last Name</th>
                <th>Nick Name</th>
            <th>Salary</th>
        </tr>
        <!— for-each processing instruction
        Looks for each element matching the XPath expression —>
        <xsl:for-each select="class/employee">
        <tr>
        <td>
            <!— value-of processing instruction
            process the value of the element matching the XPath expression
            —>
```

```
                        <xsl:value-of select = "@id"/>
                        </td>
                            <td><xsl:value-of select = "firstname"/></td>
                            <td><xsl:value-of select = "lastname"/></td>
                            <td><xsl:value-of select = "nickname"/></td>
                            <td><xsl:value-of select = "salary"/></td>
                        </tr>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Output:




```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "employee.xsl"?>
<class>
    <employee id = "001">
            <firstname>Aryan</firstname>
            <lastname>Gupta</lastname>
            <nickname>Raju</nickname>
    <salary>30000</salary>
</employee>
<employee id = "024">
```

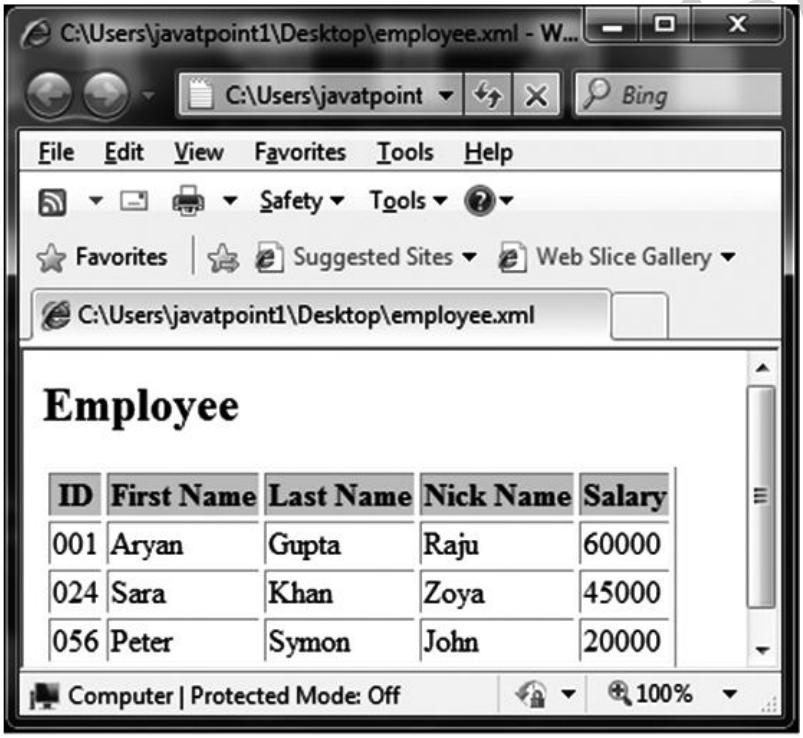
```
        <firstname>Sara</firstname>
        <lastname>Khan</lastname>
        <nickname>Zoya</nickname>
        <salary>25000</salary>
    </employee>
    <employee id = "056">
        <firstname>Peter</firstname>
        <lastname>Symon</lastname>
        <nickname>John</nickname>
        <salary>10000</salary>
    </employee>
    </class>
```

Employee.xsl

```
<?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>Employee</h2>
                        <table border = "1">
                        <tr bgcolor = "pink">
                        <th>ID</th>
                        <th>First Name</th>
                        <th>Last Name</th>
                        <th>Nick Name</th>
                        <th>Salary</th>
                    </tr>
                    <!— for-each processing instruction
                    Looks for each element matching the XPath expression —>
                    <xsl:for-each select="class/employee">
                    <tr>
                    <td>
                    <!— value-of processing instruction
                    process the value of the element matching the XPath expression
                    —>
                    <xsl:value-of select = "@id"/>
```

```
                    </td>
                    <td><xsl:value-of select = "firstname"/></td>
                    <td><xsl:value-of select = "lastname"/></td>
                <td><xsl:value-of select = "nickname"/></td>
                    <td><xsl:value-of select = "salary"/></td>
            </tr>
        </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Output



4.1.9 Document Object Model (DOM)

Q11. What is XML DOM? Explain it with example.

Ans : **(Imp.)**

DOM is an acronym stands for Document Object Model. It defines a standard way to access and manipulate documents. The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.

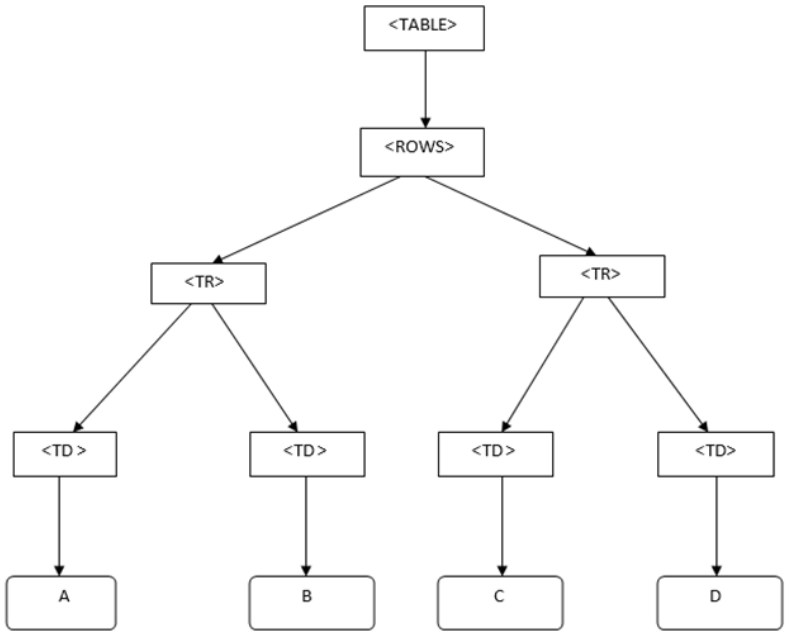
As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications. The Document Object Model can be used with any programming language.

- XML DOM defines a standard way to access and manipulate XML documents.
- The XML DOM makes a tree-structure view for an XML document.
- We can access all elements through the DOM tree.
- We can modify or delete their content and also create new elements. The elements, their content (text and attributes) are all known as nodes.

For example, consider this table, taken from an HTML document:

```
<TABLE>
<ROWS>
<TR>
<TD>A</TD>
<TD>B</TD>
</TR>
<TR>
<TD>C</TD>
<TD>D</TD>
</TR>
</ROWS>
</TABLE>
```

The Document Object Model represents this table like this:



XML DOM Example : Load XML File

Let's take an example to show how an XML document ("note.xml") is parsed into an XML DOM object.

This example parses an XML document (note.xml) into an XML DOM object and extracts information from it with JavaScript.

Let's see the note.xml file that contains message.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <note>
        <to>sonoojaiswal@javatpoint.com</to>
        <from>vimal@javatpoint.com</from>
        <body>Hello XML DOM</body>
    </note>
```

Let's see the xmldom.html file that extracts the data of XML document using DOM.

```
<!DOCTYPE html>
<html>
<body>
<h1>Important Note</h1>
<div>
<b>To:</b> <span id="to"></span><br>
<b>From:</b> <span id="from"></span><br>
<b>Message:</b> <span id="message"></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","note.xml",false);
    xmlhttp.send();
    xmlDoc=xmlhttp.responseXML;
    document.getElementById("to").innerHTML=
```

```
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
</script>
</body>
</html>
```

Output:

Important Note

To: sonoojaiswal@javatpoint.com

From: vimal@javatpoint.com

Message: Hello XML DOM

XML DOM Example : Load XML String

This example parses an XML string into an XM DOM object and then extracts some information from it with a JavaScript.

Let's see the *xmldom.html* file that extracts the data of XML string using DOM.

```
<!DOCTYPE html>
<html>
<body>
<h1>Important Note2</h1>
<div>
<b>To:</b> <span id="to"></span><br>
<b>From:</b> <span id="from"></span><br>
<b>Message:</b> <span id="message"></span>
</div>
<script>
txt1="<note>";
txt2="<to>Sania Mirza</to>";
txt3="<from>Serena William</from>";
txt4="<body>Don't forget me this weekend!</body>";
txt5="</note>";
txt=txt1+txt2+txt3+txt4+txt5;
```

```
if (window.DOMParser)
    {
    parser=new DOMParser();
    xmlDoc=parser.parseFromString(txt,"text/xml");
    }
    else // Internet Explorer
    {
    xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
    xmlDoc.async=false;
    xmlDoc.loadXML(txt);
    }
        document.getElementById("to").innerHTML=
        xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
        document.getElementById("from").innerHTML=
        xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
        document.getElementById("message").innerHTML=
        xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
    </script>
    </body>
    </html>
```

Output:

Important Note2

To: Sania Mirza

From: Serena William

Message: Don't forget me this weekend!

4.2 AJAX-ENABLED RICH INTERNET APPLICATIONS

4.2.1 Introduction

Q12. What is Ajax? What are the various technologies which supports AJAX.

Ans : **(Imp.)**

Ajax is an acronym for Asynchronous Javascript and XML. It is used to communicate with the server without refreshing the web page and thus increasing the user experience and better performance.

AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.

AJAX allows you to send only important information to the server not the entire page. So only valuable data from the client side is routed to the server side. It makes your application interactive and faster.

There are too many web applications running on the web that are using ajax technology like gmail, facebook,twitter, google map, youtube etc.

AJAX Technologies

As describe earlier, ajax is not a technology but group of inter-related technologies. AJAX technologies includes:

- HTML/XHTML and CSS

  These technologies are used for displaying content and style. It is mainly used for presentation.

- DOM

  It is used for dynamic display and interaction with data.

- XML or JSON

  For carrying data to and from server. JSON (Javascript Object Notation) is like XML but short and faster than XML.

- XMLHttpRequest

  For asynchronous communication between client and server. For more visit next page.

- JavaScript

  It is used to bring above technologies together. Independently, it is used mainly for client-side validation.

4.2.2 History of Ajax

Q13. Write the history of AJAX.

Ans :

- In the early-to-mid 1990s, most Websites were based on complete HTML pages. Each user action required a complete new page to be loaded from the server. Each time the browser reloaded a page because of a partial change, all the content had to be re-sent, even though only some of the information had changed. This placed additional load on the server and made bandwidth a limiting factor in performance.
- In 1996, the iframe tag was introduced by Internet Explorer; like the object element, it can load or fetch content asynchronously.
- In 1998, the Microsoft Outlook Web Access team developed the concept behind the XMLHttpRequest scripting object. It appeared as XMLHTTP in the second version of the MSXML library, which shipped with Internet Explorer 5.0 in March 1999
- Microsoft adopted the native XML Http Request model as of Internet Explorer 7. The ActiveX version is still supported in Internet Explorer, but not in Microsoft Edge. The utility of these background HTTP requests and asynchronous
- Google made a wide deployment of standards compliant, cross browser Ajax with Gmail (2004) and Google Maps (2005).
- In October 2004 Kayak.com's public beta release was among the first large-scale e-commerce uses of what their developers at that time called "the xml http thing". This increased interest in Ajax among web program developers.
- The term *AJAX* was publicly used on 18 February 2005 by Jesse James Garrett in an article titled *Ajax: A New Approach to Web Applications*, based on techniques used on Google pages.
- On 5 April 2006, the World Wide Web Consortium (W3C) released the first draft specification for the XMLHttpRequest object in an attempt to create an official Web standard. The latest draft of the XMLHttp Request object was published on 6 October 2016, and the XMLHttpRequest specification is now a living standard.

4.2.3 Traditional Web Applications Vs Ajax Applications

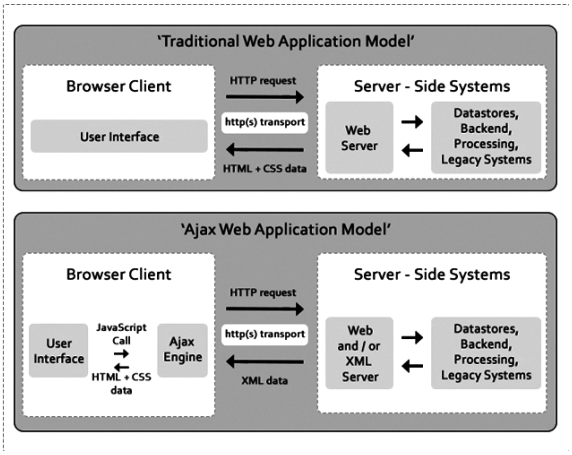
Q14. Differentiate between traditional web applications and ajax applications.

Ans : **(Imp.)**

The classic web application model works like this:

- Most user actions in the interface trigger an HTTP request back to a web server. The server does some processing retrieving data,

crunching numbers, talking to various legacy systems and then returns an HTML page to the client.



- It's a model adapted from the Web's original use as a hypertext medium, but as fans of The Elements of User Experience know, what makes the Web good for hypertext doesn't necessarily make it good for software applications.
- This approach it doesn't make for a great user experience. While the server is doing its thing, the user waiting. And at every step in a task, the user waits some more.
- Obviously, if we were designing the Web from scratch for applications, we wouldn't make users wait around. Once an interface is loaded, why should the user interaction come to a halt every time the application needs something from the server? In fact, why should the user see the application go to the server at all?

Ajax Working

- An Ajax application eliminates the start-stop-start-stop nature of interaction on the Web by introducing an intermediary an Ajax engine between the user and the server. It seems like adding a layer to the application would make it less responsive, but the opposite is true.
- Instead of loading a webpage, at the start of the session, the browser loads an Ajax engine written in JavaScript and usually tucked away in a hidden frame.
- This engine is responsible for both rendering the interface the user sees and communicating with the server on the user's behalf.
- The Ajax engine allows the user's interaction with the application to happen asynchronously - independent of communication with the server. So the user is never staring at a blank browser window and an hourglass icon, waiting around for the server to do something.

4.2.4 Rias with Ajax

Q15. Write about the importance of Rich Internet Applications with AJAX.

(OR)

What is a Rich Internet Application (RIA)?

Ans : **(Imp.)**

A rich Internet application (RIA) is a Web application designed to deliver the same features and functions normally associated with desktop applications. RIAs generally split the processing across the Internet/network divide by locating the user interface and related activity and capability on the client-side, and the data manipulation and operation on the application server-side.

An RIA normally runs inside a Web browser and usually does not require software installation on the client-side to work. However, some RIAs may only work properly with one or more specific browsers. For security purposes, most RIAs run their client portions within a special isolated area of the client desktop called as and box. The sandbox limits visibility and access to the file and operating system on the client to the application server on the other side of the connection.

This approach allows the client system to handle local activities, calculations, reformatting and so forth, thereby lowering the amount and frequency of client-server traffic, especially as compared to the client-server implementations built around so-called thin clients.

Unique Feature of RIA

One distinguishing feature of an RIA (in contrast to other Web-based applications) is the client engine that intermediates between the user and the application server. The client engine downloads when the RIA launches. The engine can be augmented during subsequent operation with additional downloads in which the engine acts as a browser extension to handle the user interface and server communications.

There are much powerful development frameworks that effectively facilitates RIA Development. Let's take a look at a few of them

AJAX

RIA performance comes from Ajax (Asynchronous JavaScript and XML), which uses client-side scripting to make web applications more responsive. Ajax applications separate client-side user interaction and server communication, and run them in parallel, reducing the delays of server-side processing normally experienced by the user.

There are many ways to implement Ajax functionality."Raw" Ajax uses JavaScript to send asynchronous requests to the server, then updates the page using the DOM. "Raw" Ajax is best suited for creating small Ajax components that asynchronously update a section of the page. These portability issues are hidden by Ajax toolkits, such as Dojo, Prototype, Script.aculo.us, and ASP.NET Ajax, which provide powerful ready-to-use controls and functions that enrich web applications and simplify JavaScript coding by making it cross-browser compatible.

Traditional web applications use XHTML forms to build simple and thin GUIs compared to the rich GUIs of Windows, Macintosh and desktop systems in general. We achieve rich GUI in RIAs with Ajax toolkits and with RIA environments such as Adobe Flex and JavaServer Faces.

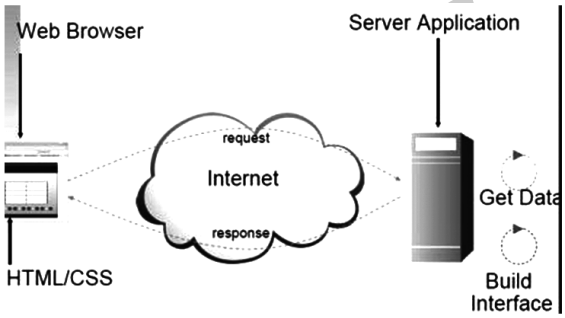
The client-side of Ajax applications is written in XHTML and CSS and uses JavaScript to add functionality to the user interface. XML is used to structure the data passed between the server and the client. We'll also use JSON (JavaScript Object Notation) for this purpose. The Ajax component that manages interaction with the server is usually implemented with JavaScript's XMLHttpRequest object—commonly abbreviated as XHR. The server processing can be implemented using any server-side technology, such as PHP, ASP. NET, JavaServer Faces and Ruby on Rails.

Rich Internet Applications (RIA) & AJAX

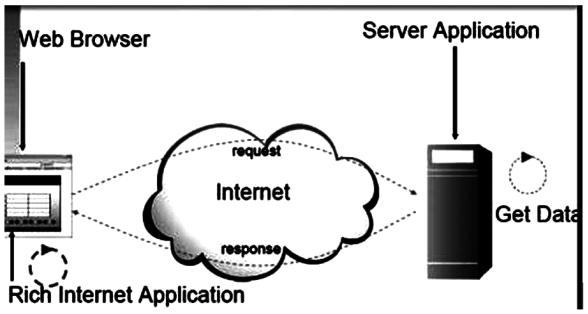
The framework in which we can find AJAX is RIA – Rich Internet Application

Rich means if it is internet application then it is running in web browser and this has two features. It provides sophisticated UI and also provides communication to backend service. The follwing diagrams show the different between legacy applications and RIA applications.

Legacy application



RIA applications



Non rich clients are thin clients because web browser didn't do any kind of processing. Only server do processing and generates html response. Browser only shows response

In Rich applications the clients are not THIN because some processing is done by server and rest of it is done in browser. RICH Applications provides more Interactivity, distributed Processing and greater performance. User has control of the application.

Some of the RIA Technologies are

1. Ajax (Dojo, GWT, Prototype, etc.)
2. Adobe Flex
3. Curl
4. Microsoft Silverlight
5. FLASH
6. Mozilla XUL
7. Applets

4.2.5 Ajax Example Using XMLHTTPRequest Object

Q16. What is XMLHttpRequest Object? How to create an XMLHttpRequest Object.

Ans : **(Imp.)**

The AJAX – The XMLHttpRequest object is an API that is utilized for retrieving data from a specific server. AJAX programming makes extensive use of the XMLHttpRequest. It can fetch any kind of data including text, XML, JSON. In the background, the XMLHttpRequest Object requests for the data and then updates the website without requiring the client to reload the page. To maintain the asynchronous communication between server and client, an object of type XMLHttpRequest Object is needed.

How to create an XMLHttpRequest Object

A built-in XMLHttpRequest object is avialble in all of the modern browsers such as Edge, Chrome, Firefox, Opera, and Safari. To create an XMLHttpRequest object, you have to follow the below-given syntax of XMLHttpRequest Object:

```
var variableName = new XMLHttpRequest();
```

Example: Using AJAX – The XMLHttpRequest Object

In this example, we will try to fetch the content of the **"ajax_info.txt"** file from our server, and then we will replace the paragraph content with it:

```
<!DOCTYPE html>
<html>
<body>
<h1>AJAX - The XMLHttpRequest Object | Explained</h1>
```

Here, we have added a paragraph with the tag and a **"Change paragraph"** button which we will invoke the **"loadingDoc()"** function, when we will click this button:

```
<p id="p1">We will change this paragraph.</p>
<button type="button" onclick="loadingDoc()">Change paragraph</button>
<script>
```

In the loadingDoc() function, firstly, we will add a **"xhttp"** XMLHttpRequest object:

```
function loadingDoc() {
var xhttp = new XMLHttpRequest();
```

Next, we will add an event handler **"onreadystatechange"** which will be invoked whenever the **"readyState"** attribute changes its value. If the request has been sent and the received response signifies that request has been succeeded, then it will be written in our HTML paragraph element:

```
    xhttp.onreadystatechange = function() {
                if (this.readyState == 4 && this.status == 200) {
            document.getElementById("p1").innerHTML = this.responseText;
    }
};
```

The **"xhttp"** XMLHttpRequest Object will get the **"ajax_info.txt"** file from the server and then with the help of the **"send()"** method, it will send the request to the server:

```
    xhttp.open("GET", "ajax_info.txt", true);
    xhttp.send();
}
</script>
</body>
</html>
```

Execute the above-given program in your favorite code editor or any online coding sandbox; however, we will utilize the JSBin for this purpose:

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

<h1>AJAX - The XMLHttpRequest Object | Explained</h1>

<p id="p1">We will change this paragraph.</p>

<button type="button" onclick="loadingDoc()">Change paragraph</button>

<script>
function loadingDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("p1").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

4.2.6 XML AND DOM

Q17. How AXAX can be communicated with XML file? Explain.

Ans : **(Imp.)**

AJAX can be used for interactive communication with an XML file.

AJAX XML Example

The following example will demonstrate how a web page can fetch information from an XML file with AJAX:

When a user clicks on the "Get CD info" button above, the loadDoc() function is executed.

The loadDoc() function creates an XMLHttpRequest object, adds the function to be executed when the server response is ready, and sends the request off to the server.

When the server response is ready, an HTML table is built, nodes (elements) are extracted from the XML file, and it finally updates the element "demo" with the HTML table filled with XML data:

LoadXMLDoc()

```
function loadDoc() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
        myFunction(this);
        }
    };
    xhttp.open("GET", "cd_catalog.xml", true);
    xhttp.send();
    }
    function myFunction(xml) {
    var i;
    var xmlDoc = xml.responseXML;
    var table="<tr><th>Title</th><th>Artist</th></tr>";
    var x = xmlDoc.getElementsByTagName("CD");
    for (i = 0; i <x.length; i++) {
        table += "<tr><td>" +
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue + "</td><td>" +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue + "</td></tr>";
    }
    document.getElementById("demo").innerHTML = table;
    }
```

The XML File

The XML file used in the example above looks like this: "cd_catalog.xml".

Q18. Explain about Document Object Model in AJAX.

Ans :

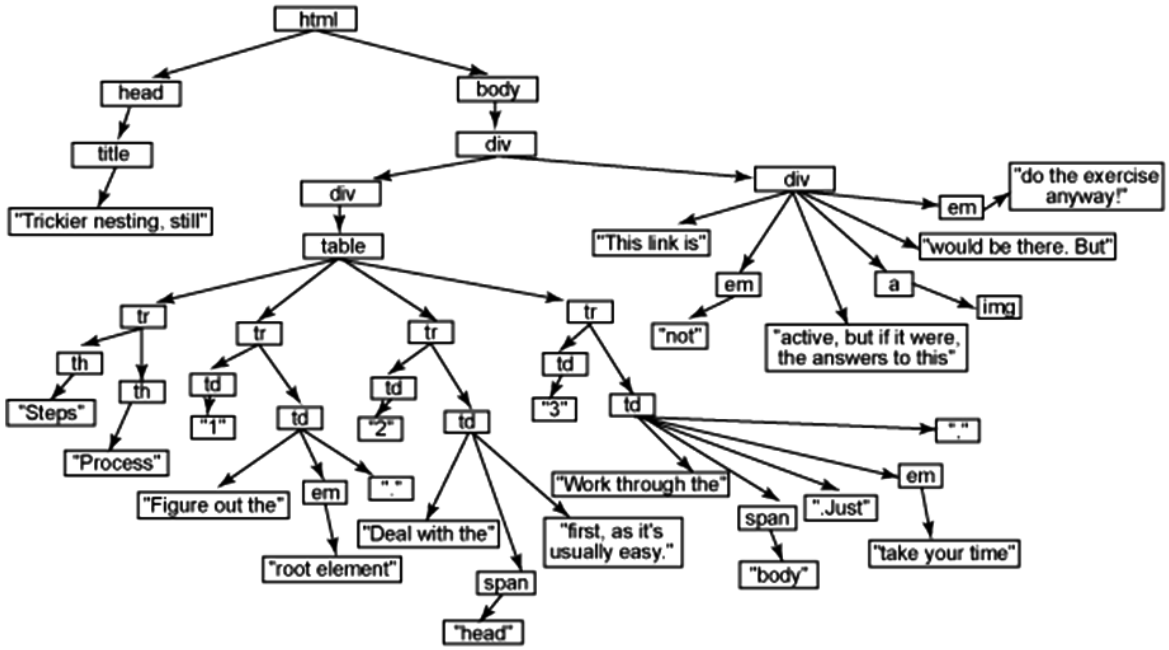
DOM – Document Object Model

The Document Object Model is a W3C standard .Because of that, all modern Web browsers support the DOM. The DOM is also a cross-language specification; in other words, you can use it from most of the popular programming languages

The HTML web pages are all about the organization of the content on the page. Markup is all about providing this level of organization; a p indicates that text is in a paragraph, img denotes an image, div divides a page up into sections, and so forth. The browser takes all this textual organization and turns it into something much more interesting — a set of objects, each of which can be changed, added to, or deleted. Today's web browsers change the markup language pages into an object tree structure.

For example the below given HTML page is converted to the shown tree structure:

```
<html>
<head>
<title>Trickier nesting, still</title>
</head>
<body>
<div id="main-body">
<div id="contents">
<table>
<tr><th>Steps</th><th>Process</th></tr>
<tr><td>1</td><td>Figure out the <em>root
element</em>.</td></tr>
<tr><td>2</td><td>Deal with the <span id="code">head</span>
first,
as it's usually easy.</td></tr>
<tr><td>3</td><td>Work through the <span id="code">body</span>.
Just <em>take your time</em>.</td></tr>
</table>
</div>
<div id="closing">
This link is <em>not</em> active, but if it were, the answers
to this <a href="answers.html"><img src="exercise.gif" /></a>
would be there. But <em>do the exercise anyway!</em>
</div>
</div>
</body>
</html>
```



The above given tree structure is the DOM tree notation of the HTML page. DOM stands for Document Object Model and is a specification available from the World Wide Web Consortium. More importantly though, the DOM defines the objects' types and properties that allow a browser to represent markup.

Node

A node is the most basic object type in the DOM. In fact, almost every other object defined by the DOM extends the node object. In a DOM tree, almost everything is a node. Every element is at its most basic level a node in the DOM tree. Every attribute is a node. Every piece of text is a node.

According to the DOM, everything in an HTML document is a node.The DOM says that:

- The entire document is a document node
- Every HTML tag is an element node
- The texts contained in the HTML elements are text nodes
- Every HTML attribute is an attribute node
- Comments are comment nodes

Properties of a node

The key properties of a DOM node are:

- nodeName reports the name of the.
- nodeValue gives the "value" of the node.
- nodeType gives the type of node(element node, attribute node, text node or document node)
- parentNode returns the node's parent.
- childNodes is a list of a node's children.
- firstChild is just a shortcut to the first node in the childNodes list.

- lastChild is another shortcut, this time to the last node in the childNodes list.
- previousSibling returns the node before the current node. In other words, it returns the node that precedes the current one, in this node's parent's childNodes list .
- nextSibling is similar to the previousSibling property; it turns the next node in the parent's childNodes list.
- attributes is only useful on an element node; it returns a list of an element's attributes.

Methods of a node

Next up are the methods available to all nodes:

- **insertBefore(newChild, referenceNode)** inserts the newChild node before the referenceNode.
- **replaceChild(newChild, oldChild)** replaces the oldChild node with the newChild node.
- **removeChild(oldChild)** removes the oldChild node from the node the function is run on.
- **appendChild(newChild)** adds the newChild node to the node this function is run on.
- **newChild** is added at the end of the target node's children.
- **hasChildNodes()** returns true if the node it's called on has children, and false if it doesn't.
- **hasAttributes()** returns true if the node it's called on has attributes, and false if there are no attributes.

4.2.7 Creating full Scale Ajax - Enabled Application

Q19. Explain, how to create full scale AJAX application.

Ans : **(Imp.)**

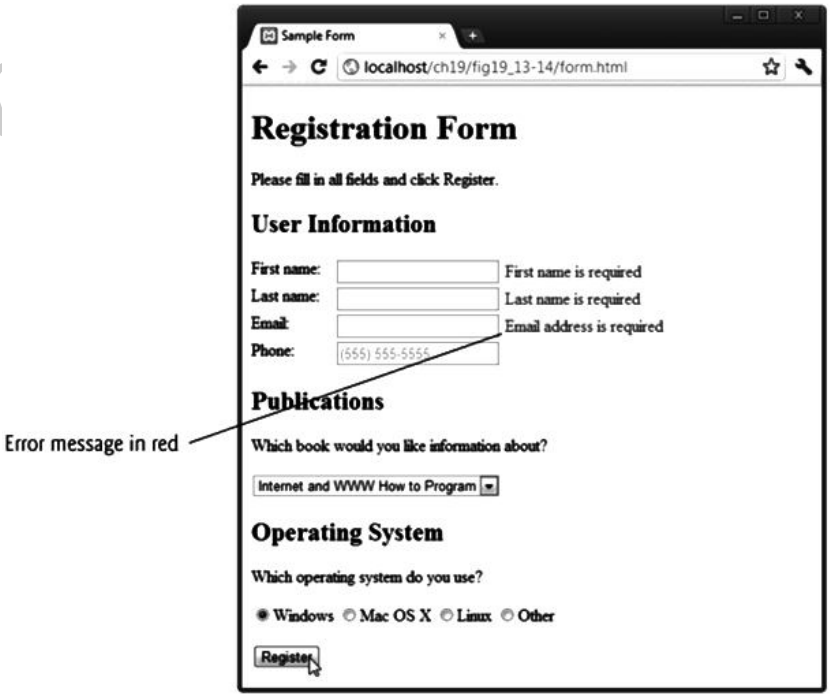
Rich Internet Applications (RIAs) with Ajax

- Classic HTML registration form
  - Sends all of the data to be validated to the server when the user clicks the Register button
  - While the server is validating the data, the user cannot interact with the page
  - Server finds invalid data, generates a new page identifying the errors in the form and sends it back to the client-which renders the page in the browser
  - User fixes the errors and clicks the Register button again
  - Cycle repeats until no errors are found, then the data is stored on the server
  - Entire page reloads every time the user submits invalid data
- Ajax-enabled forms are more interactive
  - Entries are validated individually, dynamically as the user enters data into the fields
  - If a problem is found, the server sends an error message that is asynchronously displayed to inform the user of the problem
  - Sending each entry asynchronously allows the user to address invalid entries quickly, rather than making edits and resubmitting the entire form repeatedly until all entries are valid
  - Asynchronous requests could also be used to fill some fields based on previous fields' values (e.g., city based on zip code)

a) A sample registration form in which the user has not filled in the required fields, but attempts to submit the form anyway by clicking Register

Fig.: Classic HTML5 form: The user submits the form to the server, which validates the data (if any). Server responds indicating any fields with invalid or missing data

b) The server responds by indicating all the form fields with missing or invalid data. The user must correct the problems and resubmit the entire form repeatedly until alt errors are corrected.



Creating a Full-Scale Ajax-Enabled Application

- The web application may interacts with a web service to obtain data and to modify data in a server-side database.
- The web application and server communicate with a data format called JSON (JavaScript Object Notation).
- The application executes server-side functions (or APIs) that occurs in parallel with the user interacting with the web application.

Using JSON

- JSON (JavaScript Object Notation)
  - Simple way to represent JavaScript objects as strings
  - A simpler alternative to XML for passing data between the client and the server
  - API in many programming languages
- JSON object
  - Represented as a list of property names and values contained in curly braces
  - {“propertyName1” : value1, “property Name2”: value2 }
- Array
  - Represented in JSON with square brackets containing a comma-separated list of values
  - Each value in a JSON array can be a string, a number, a JSON representation of an object, true, false or null
  - “array_name”: [{“propertyName1” : value1, “propertyName2”: value2}, {“propertyName1” : value1, “propertyName2”: value2}]

- JSON strings
  - Easier to create and parse than XML
  - Require fewer bytes
  - For these reasons, JSON is commonly used to communicate in client/server interaction

Parsing JSON Data

- Web service's functions or APIs may returns a JSON representation of an object or array of objects.
- For example, when the web application requests the list of names in the address book, the list is returned as a JSON array.
- Address-book data formatted in JSON:

  [{ "first": "Cheryl", "last": "Black" },

  { "first": "James", "last": "Blue" },

  { "first": "Mike", "last": "Brown" },

  { "first": "Meg", "last": "Gold" }]
- When the XMLHttpRequest object receives the response, it calls the JSON.parse function, which converts the JSON string into a JavaScript object.

4.2.8 DOJO TOOLKIT

Q20. Define Dojo Toolkit. Explain.

Ans :

The Dojo toolkit is an open-source modular toolkit containing a JavaScript library that is designed for rapidly creating JavaScript/Ajax-based websites and cross-platform applications. It aims to save time and scale the development process by using the Web standards themselves as the platform. One big advantage of the Dojo toolkit is that its core is a set of lightweight and independent modules that can be loaded as needed asynchronously and very quickly.

The Dojo toolkit is also known simply as Dojo.

DOJO is being used by all the popular internet browsers like Internet Explorer, Google Chrome, Safari, Firefox, and Opera and on smart phones and tablets by Apple (iPhone, iPod) Google (Android) and BlackBerry.

DOJO provides us many widgets, utilities and AJAX libraries to develop our applications.

DOJO provides the connection between DOM element and DOJO function and also contains many DOM elements looks like,

1. HTML,
2. SVG and
3. Style packages.

DOJO is helping us to create rich and interactive web applications and it is similar like Comet and Ajax.

Dojo has great support for integrating with back end data store. It works seamlessly with **REST services**. We even extended Dojo's JSON-Rest-Store for adding custom encoding or decoding. It worked like a **charm**.

DOJO is a great framework for developing **Ajax** based web applications.

It is also helping to the programmers to develop powerful web apps very easily and less time.

We can downloaded the DOJO Toolkit from http://dojotoolkit.org/

Features of Dojo?

- Dojo is an open source JavaScript toolkit.
- It is easy to learn.
- It is used to develop highly interactive web applications.
- It offers widgets, utilities, and higher IO abstraction.
- It is licensed by BSD or AFL.

The basic directory structure of Dojo is simple and contains the following three points:

/index.html: The application entry point.

/app: The application module.

/app/main.js: The main script for the app module.

Advantages of DOJO:

Dojo is a very high-quality JavaScript toolkit. It has several advantages or benefits that support:

- Loosely typed variables
- Associative arrays
- Objects and classes
- W3C DOM support in the Dojo
- Regular expression
- Associative arrays

Disadvantages/ limitation of Dojo:

- Dojo supports limited browsers.
- You can't hide the Dojo codes in the case of commercial application.
- Dojo requires many networks.
- Documentation is quite narrow.

Short Question & Answers

1. XML.

Ans :

- XML (extensible Markup Language) is a mark up language.
- XML is designed to store and transport data.
- Xml was released in late 90's. it was created to provide an easy to use and store self describing data.
- XML became a W3C Recommendation on February 10, 1998.
- XML is not a replacement for HTML.
- XML is designed to be self-descriptive.
- XML is designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is platform independent and language independent.

2. Features of XML.

Ans :

XML is widely used in the era of web development. It is also used to simplify data storage and data sharing.

The main features or advantages of XML are given below.

1. XML separates data from HTML

If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes.

With XML, data can be stored in separate XML files. This way you can focus on using HTML/CSS for display and layout, and be sure that changes in the underlying data will not require any changes to the HTM

With a few lines of JavaScript code, you can read an external XML file and update the data content of your web page.

2. XML simplifies data sharing

In the real world, computer systems and databases contain data in incompatible formats.

XML data is stored in plain text format. This provides a software- and hardware indepen-dent way of storing data.

This makes it much easier to create data that can be shared by different applications.

3. XML simplifies data transport

One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet.

Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

4. XML simplifies Platform change

Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost.

XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

5. XML increases data availability

Different applications can access your data, not only in HTML pages, but also from XML data sources.

With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.

6. XML can be used to create new internet languages

A lot of new Internet languages are created with XML.

Here are some examples:

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

3. Explain the structure of XML data.

Ans :

In XML document has a self descriptive structure. It forms a tree structure which is referred as an XML tree. The tree structure makes easy to describe an XML document.

A tree structure contains root element (as parent), child element and so on. It is very easy to traverse all succeeding branches and sub-branches and leaf nodes starting from the root.

1. An XML document contains exactly one root element: the start tag of the XML document, and it contains all other elements.

```
<root>
    <section>
        <sub-section></sub-section>
        <sub-section></sub-section>
    </section>
    <section>
        <sub-section></sub-section>
        <sub-section></sub-section>
```

```
</section>
<root>
```

2. XML documents may begin with a prolog that appears before the root element. It has the metadata about the XML document, such as character encoding, document structure, and style sheets. For example,

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

3. A tag in XML is a case-sensitive markup construct that begins with < and ends with >. A tag can be:

- A start-tag, such as <name>
- An end-tag, such as </name>
- An empty-element tag, such as <name>

4. XML Namespaces.

Ans :

XML Namespace is used to avoid element name conflict in XML document.

XML Namespace Declaration

An XML namespace is declared using the reserved XML attribute. This attribute name must be started with "xmlns".

syntax: <element xmlns:name = "URL">

Here, namespace starts with keyword "xmlns". The word name is a namespace prefix. The URL is a namespace identifier.

Let's see the example of XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<cont:contact xmlns:cont="http://sssit.org/contact-us">
<cont:name>Vimal Jaiswal</cont:name>
<cont:company>SSSIT.org</cont:company>
<cont:phone>(0120) 425-6464</cont:phone>
</cont:contact>
```

Namespace Prefix: cont

Namespace Identifier: http://sssit.org/contact-us

It specifies that the element name and attribute names with cont prefix belongs to http://sssit.org/contact-us name space.

In XML, elements name are defined by the developer so there is a chance to conflict in name of the elements. To avoid these types of confliction we use XML Namespaces. We can say that XML Namespaces provide a method to avoid element name conflict.

Generally these conflict occurs when we try to mix XML documents from different XML application.

Let's take an example with two tables:

Table1

```
<table>
    <tr>
        <td>Aries</td>
        <td>Bingo</td>
    </tr>
</table>
```

Table2: This table carries information about a computer table.

```
<table>
    <name>Computer table</name>
    <width>80</width>
    <length>120</length>
</table>
```

If you add these both XML fragments together, there would be a name conflict because both have <table< element. Although they have different name and meaning.

5. XML validation.

Ans :

A well formed XML document can be validated against DTD or Schema.

A well-formed XML document is an XML document with correct syntax. It is very necessary to know about valid XML document before knowing XML validation.

Valid XML document

- It must be well formed (satisfy all the basic syntax condition)
- It should be behave according to predefined DTD or XML schema

Rules for well formed XML

- It must begin with the XML declaration.
- It must have one unique root element.
- All start tags of XML documents must match end tags.
- XML tags are case sensitive.
- All elements must be closed.
- All elements must be properly nested.
- All attributes values must be quoted.
- XML entities must be used for special characters.

6. What is XML schema? Explain.

Ans :

XML schema is a language which is used for expressing constraint about XML documents. There are so many schema languages which are used now a days for example Relax- NG and XSD (XML schema definition).

An XML schema is used to define the structure of an XML document. It is like DTD but provides more control on XML structure.

Checking Validation

An XML document is called "well-formed" if it contains the correct syntax. A well-formed and valid XML document is one which have been validated against Schema.

XML Schema Example

Let's create a schema file *employee.xsd*

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.javatpoint.com"
xmlns="http://www.javatpoint.com"
elementFormDefault="qualified">
<xs:element name="employee">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="firstname" type="xs:string"/>
            <xs:element name="lastname" type="xs:string"/>
            <xs:element name="email" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Let's see the xml file using XML schema or XSD file.

employee.xml

```
<?xml version="1.0"?>
<employee
xmlns="http://www.javatpoint.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.javatpoint.com employee.xsd">
<firstname>vimal</firstname>
<lastname>jaiswal</lastname>
<email>vimal@javatpoint.com</email>
</employee>
```

7. XML vocabularies.

Ans :

XML vocabularies are the elements used in particular applications or data formats - the definitions of the meanings of those formats. For example, in CDF, element names such as <SCHEDULE>, <CHANNEL>, and <ITEM> make up the vocabulary for describing collections of pages, when these pages should be downloaded, etc.

Vocabularies, along with the structural relationships between the elements, are defined in XML DTDs or XML schemas.

XML elements can be defined as building blocks of an XML. Elements can behave as containers to hold text, elements, attributes, media objects or all of these.

Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

Syntax

<element-nameattribute1attribute2>

....content

</element-name>

where,

- **element-name** is the name of the element. The *name* its case in the start and end tags must match.
- **attribute1, attribute2** are attributes of the element separated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as "name = "value" *name* is followed by an = sign and a string *value* inside double(" ") or single(' ') quotes.

Empty Element

An empty element (element with no content) has following syntax "

<nameattribute1attribute2.../>

Following is an example of an XML document using various XML element "

<?xml version = "1.0"?>

<contact-info>

<addresscategory = "residence">

<name>Tanmay Patil</name>

<company>TutorialsPoint</company>

<phone>(011) 123-4567</phone>

</address>

</contact-info>

8. What is XSL?

Ans :

In HTML documents, tags are predefined but in XML documents, tags are not predefined. World Wide Web Consortium (W3C) developed XSL to understand and style an XML document, which can act as XML based Stylesheet Language.

An XSL document specifies how a browser should render an XML document.

Main parts of XSL Document

- **XSLT:** It is a language for transforming XML documents into various other types of documents.
- **XPath:** It is a language for navigating in XML documents.
- **XQuery:** It is a language for querying XML documents.
- **XSL-FO:** It is a language for formatting XML documents.

9. Advantage of XSLT

Ans ;

- XSLT provides an easy way to merge XML data into presentation because it applies user defined transformations to an XML document and the output can be HTML, XML, or any other structured document.
- XSLT provides Xpath to locate elements/attribute within an XML document. So it is more convenient way to traverse an XML document rather than a traditional way, by using scripting language.
- XSLT is template based. So it is more resilient to changes in documents than low level DOM and SAX.
- By using XML and XSLT, the application UI script will look clean and will be easier to maintain.
- XSLT templates are based on XPath pattern which is very powerful in terms of performance to process the XML document.
- XSLT can be used as a validation language as it uses tree-pattern-matching approach.
- You can change the output simply modifying the transformations in XSL files.

10. What is XML DOM?

Ans :

DOM is an acronym stands for Document Object Model. It defines a standard way to access and manipulate documents. The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.

As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications. The Document Object Model can be used with any programming language.

- XML DOM defines a standard way to access and manipulate XML documents.
- The XML DOM makes a tree-structure view for an XML document.
- We can access all elements through the DOM tree.

- We can modify or delete their content and also create new elements. The elements, their content (text and attributes) are all known as nodes.

For example, consider this table, taken from an HTML document:

```
<TABLE>
<ROWS>
<TR>
<TD>A</TD>
<TD>B</TD>
</TR>
<TR>
<TD>C</TD>
<TD>D</TD>
</TR>
</ROWS>
</TABLE>
```

11. What is Ajax?

Ans :

Ajax is an acronym for Asynchronous Javascript and XML. It is used to communicate with the server without refreshing the web page and thus increasing the user experience and better performance.

AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.

AJAX allows you to send only important information to the server not the entire page. So only valuable data from the client side is routed to the server side. It makes your application interactive and faster.

There are too many web applications running on the web that are using ajax technology like gmail, facebook,twitter, google map, youtube etc.

12. Write the history of AJAX.

Ans :

- In the early-to-mid 1990s, most Websites were based on complete HTML pages. Each user action required a complete new page to be loaded from the server. Each time the browser reloaded a page because of a partial change, all the content had to be re-sent, even though only some of the information had changed. This placed additional load on the server and made bandwidth a limiting factor in performance.
- In 1996, the iframe tag was introduced by Internet Explorer; like the object element, it can load or fetch content asynchronously.
- In 1998, the Microsoft Outlook Web Access team developed the concept behind the XMLHttpRequest scripting object. It appeared as XMLHTTP in the second version of the MSXML library, which shipped with Internet Explorer 5.0 in March 1999

- Microsoft adopted the native XML Http Request model as of Internet Explorer 7. The ActiveX version is still supported in Internet Explorer, but not in Microsoft Edge. The utility of these background HTTP requests and asynchronous
- Google made a wide deployment of standards compliant, cross browser Ajax with Gmail (2004) and Google Maps (2005).
- In October 2004 Kayak.com's public beta release was among the first large-scale e-commerce uses of what their developers at that time called "the xml http thing". This increased interest in Ajax among web program developers.
- The term *AJAX* was publicly used on 18 February 2005 by Jesse James Garrett in an article titled *Ajax: A New Approach to Web Applications*, based on techniques used on Google pages.

13. What is a Rich Internet Application?

Ans :

A rich Internet application (RIA) is a Web application designed to deliver the same features and functions normally associated with desktop applications. RIAs generally split the processing across the Internet/network divide by locating the user interface and related activity and capability on the client-side, and the data manipulation and operation on the application server-side.

An RIA normally runs inside a Web browser and usually does not require software installation on the client-side to work. However, some RIAs may only work properly with one or more specific browsers. For security purposes, most RIAs run their client portions within a special isolated area of the client desktop called as and box. The sandbox limits visibility and access to the file and operating system on the client to the application server on the other side of the connection.

This approach allows the client system to handle local activities, calculations, reformatting and so forth, thereby lowering the amount and frequency of client-server traffic, especially as compared to the client-server implementations built around so-called thin clients.

14. Define Dojo Toolkit. Explain.

Ans :

The Dojo toolkit is an open-source modular toolkit containing a JavaScript library that is designed for rapidly creating JavaScript/Ajax-based websites and cross-platform applications. It aims to save time and scale the development process by using the Web standards themselves as the platform. One big advantage of the Dojo toolkit is that its core is a set of lightweight and independent modules that can be loaded as needed asynchronously and very quickly.

The Dojo toolkit is also known simply as Dojo.

DOJO is being used by all the popular internet browsers like Internet Explorer, Google Chrome, Safari, Firefox, and Opera and on smart phones and tablets by Apple (iPhone, iPod) Google (Android) and BlackBerry.

DOJO provides us many widgets, utilities and AJAX libraries to develop our applications.

DOJO provides the connection between DOM element and DOJO function and also contains many DOM elements looks like,

1. HTML,
2. SVG and
3. Style packages.

Choose the Correct Answer

1. What does XML stand for? [b]

   (a) eXtra Modern Link (b) eXtensibleMarkup Language

   (c) ExampleMarkup Language (d) X-Markup Language

2. DTD Stands for _____. [a]

   (a) Document Type Definition (b) Document Transport Direction

   (c) Document Transcript Definition (d) None of the above

3. Schema data type can be specified using [a]

   (a) Dt:type (b) Data Type

   (c) XLink (d) All of the above

4. Which Of The Following Strings Are A Correct XML Name? [b]

   (a) My Element (b) _myElement

   (c) #myElement (d) All of the above

5. What is the correct syntax of the declaration which defines the XML version? [b]

   (a) <?xml version="A.0"?> (b) <xml version="A.0" />

   (c) <?xml version="A.0" /> (d) None of the above

6. Which Internet Language Is Used For Describing Available Web Services In XML? [c]

   (a) RSS (b) RDF

   (c) WSDL (d) OWL

7. Which of the following is a valid XSLT iteration command [b]

   (a) for-all (b) for each

   (c) for (d) in-turn

8. Ajax is used for creating _____. [a]

   (a) Web applications (b) Desktop applications

   (c) System applications (d) Both A. and B.

9. What server support Ajax? [c]

(a) WWW (b) SMTP

(c) HTTP (d) All of the above

10. Which syntax is used to insert comments into an XML document? [c]

(a) <comment>This is a comment</comment>

(b) <?-This is a comment->

(c) <!-This is a comment->

(d) <- -This is a comment- ->

Fill in the blanks

1. XML is designed to _____ and store data.
2. WSDL stands for _____.
3. DOM stands for _____.
4. The XSL formatting object which formats the data in a table _____.
5. Ajax stands for _____.
6. _____ provides the ability to asynchronously exchange data between Web browsers and a Web server.
7. The combination of _____ and _____ give Ajax its name.
8. The _____ is an open-source modular toolkit containing a JavaScript library that is designed for rapidly creating JavaScript/Ajax-based websites and cross-platform applications.
9. A _____ is the most basic object type in the DOM
10. RIA full form _____.

ANSWERS

1. Transport
2. Web Services Description Language
3. Document Object Model
4. Table
5. Asynchronous JavaScript and XML
6. XMLHttpRequest object accomplish in Ajax
7. Asynchronous JavaScript and XML
8. Dojo toolkit
9. node
10. Rich internet Applications

Lab Practicals

**Q1. Write a HTML program using basic text formatting tags, <p>,
, <pre>.**

Ans:

```
<html>
<body>
<h2>Example of PRE, P, BR tags</h2>
<!— Use of <p> tag —>
<p>This paragraph
contains a lot of lines in the source code, but the browser ignores it.</p>
<br>
<br>
<pre>
Text in a pre element is displayed in a fixed-width font, and it preserves both spaces and line breaks
</pre>
<br>
</body>
</html>
```

Q2. Write a HTML program by using text formatting tags.

Ans:

```
<html>
<body>
<h2>Example </h2>
```

```
<p>This is <b>bold text</b>.</p>
<p>This is <strong>strongly important text</strong>.</p>
<p>This is <i>italic text</i>.</p>
<p>This is <em>emphasized text</em>.</p>
<p>This is <mark>highlighted text</mark>.</p>
<p>This is <code>computer code</code>.</p>
<p>This is <small>smaller text</small>.</p>
<p>This is <sub>subscript</sub> and <sup>superscript</sup> text.</p>
<p>This is <del>deleted text</del>.</p>
<p>This is <ins>inserted text</ins>.</p>
<br>
</body>
</html>
```

Q3. Write a HTML program using presentational element tags <b>, <i>, <strike>, <sup>, <sub>, <big>, <small>, <hr>.

Ans :

```
<html>
<body>
<h2>Example </h2>
<b>This text in bold </b>
<i>This text in Italic </i>
<strike> This is striked out </strike>
<br>
```

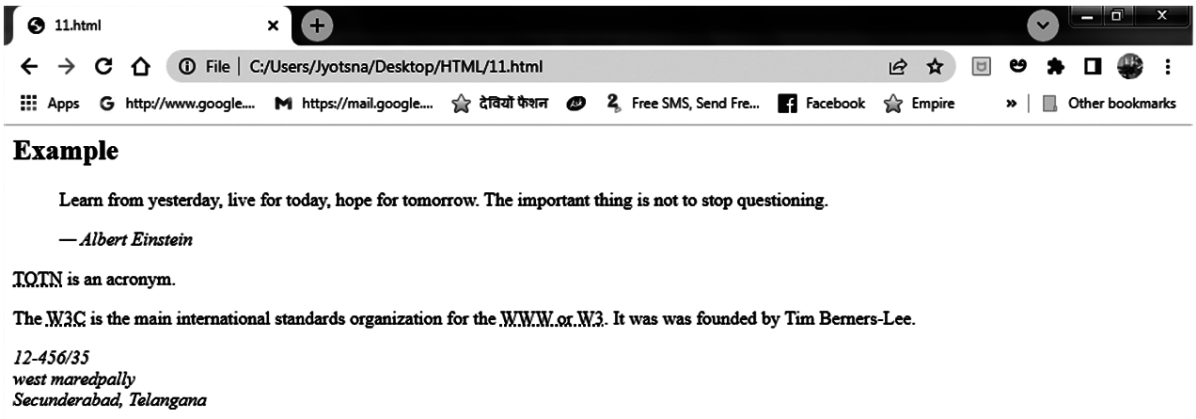
```
This is ruled
<hr>
<sup> This is superscipted</sup>
<sub> This is subscipted</sub>
<big> This text is big </big>
<small> This text is small </small>
<br>
</body>
</html>
```

Q4. Write a HTML program using phrase element tags <blockquote>, <cite>, <abbr>, <acronym>, <kbd>, <address>.

Ans :

```
<html>
<body>
<h2>Example </h2>
<blockquote>
    <p>Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop questioning.</p>
<cite>— Albert Einstein</cite>
</blockquote>
<p> <acronym title="Tech On The Net">TOTN</acronym> is an acronym.</p>
    <p>The <abbr title="World Wide Web Consortium">W3C</abbr> is the main international standards organization for the <abbr title="World Wide Web">WWW or W3</abbr>.
It was was founded by Tim Berners-Lee.</p>
<address>
```

```
        12-456/35<br>
        westmaredpally<br>
        Secunderabad, Telangana
        </address>
<br>
</body>
</html>
```



Q5. Write a HTML program using different list types.

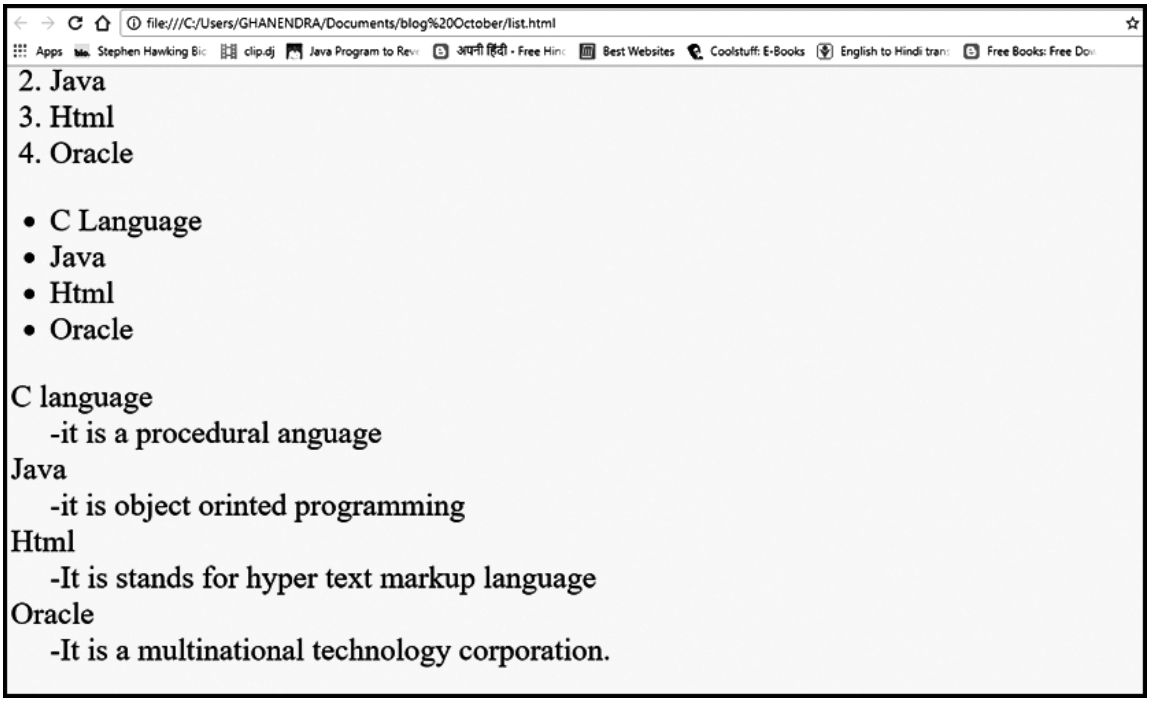
Ans :

```
<html>
<head>
<title>Type Of List</title>
</head>
<body bgcolor="green" >
<font size="18">
<ol>
    <li>C Language</li>
    <li>Java</li>
    <li>Html</li>
    <li>Oracle</li>
</ol>
<ul>
<li>C Language</li>
<li>Java</li>
<li>Html</li>
<li>Oracle</li>
```

```
</ul>
<dl>
<dt>C language</dt>
<dd>-it is a procedural anguage</dd>
<dt>Java</dt>
<dd>-it is object oriented programming</dd>
<dt>Html</dt>
<dd>-It is  stands for hyper textmarkup language</dd>
<dt>Oracle</dt>
<dd>-It is a multinational technology corporation.</dd>
</dl>
</font >
</body>
</html>
```

Copy Ends Here

Output



Q6. Create a HTML page that displays ingredients and instructions to prepare a recipe.

Ans :

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

<title>Simple caprese salad</title>

</head>

<body>

<header>

<section>

<h2>Simple caprese salad</h2>

A spin on the standard caprese salad.

</section>

<section>

Traditionally, the caprese salad calls for sliced tomatoes and slices of mozzarella. This version uses cherry tomatoes, and either mozzarella balls or small cubes of the cheese. This gives you a salad that's a bit more like a salad, and works better as a side item. In addition, the cherry tomatoes are a bit sweeter, which brings a little more brightness to the dish.

The recipe below is designed to act as a guide, and not something to be followed prescriptively. Play with your food! Adjust amounts based on your tastes.

</section>

</header>

<article>

<section id="ingredients">

<h3>Ingredients</h3>

Serves 2 people

10 cherry tomatoes, sliced in half

3 oz fresh mozzarella, cut to size to match cherry tomatoes

6 basil leaves, chiffonade

2 Tbsp extra virgin olive oil, or to taste

1 Tbsp balsamic vinegar, or to taste

2 healthy pinches of Kosher salt, or to taste

6-8 grinds of pepper, or to taste

</section>

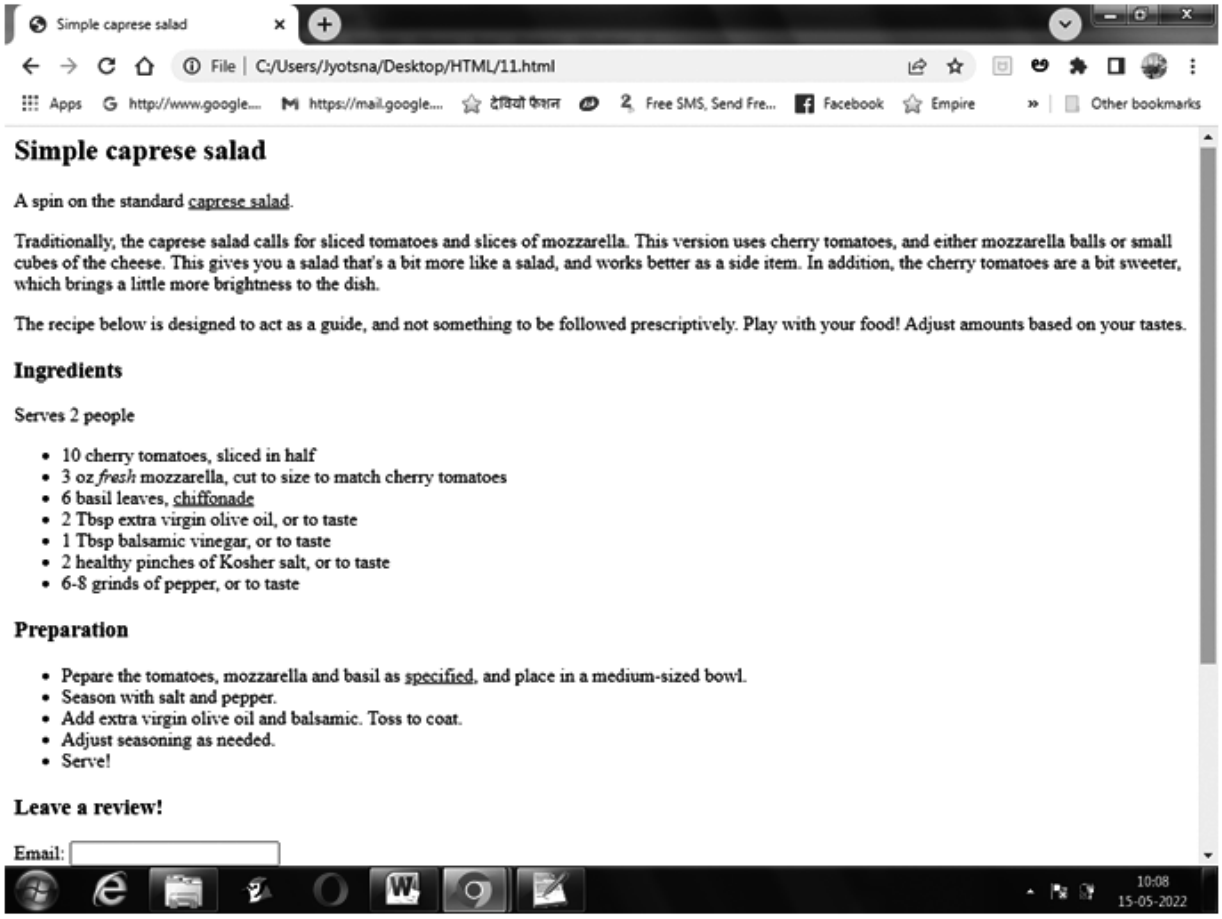
<section>

<h3>Preparation</h3>

Pepare the tomatoes, mozzarella and basil as specified, and place in a medium-sized

```
        bowl.</li>
<li>Season with salt and pepper.</li>
<li>Add extra virgin olive oil and balsamic. Toss to coat.</li>
<li>Adjust seasoning as needed.</li>
<li>Serve!</li>
</ul>
</section>
</article>
<footer>
<section>
<h3>Leave a review!</h3>
<form>
<div>
<label for="email">Email:</label>
<input type="email" id="email" />
</div>
<div>
<label for="review">Review:</label>
<textarea id="review" cols="60" rows="5"></textarea>
</div>
</form>
</section>
<section>
<h3>Reviews</h3>
<table>
<thead>
<tr>
<th>Email</th>
<th>Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>techie4good@adventure-works.com</td>
<td>Loved it!!</td>
</tr>
<tr>
```

```
<td>geektrainer@adventure-works.com</td>
<td>Used extra cheese. Came out great!</td>
</tr>
</tbody>
</table>
</section>
</footer>
</body>
</html>
```



Q7. Write a HTML program using grouping elements and .

Ans :

```
<html>
<head>
<title>HTML tutorial</title>
</head>
```

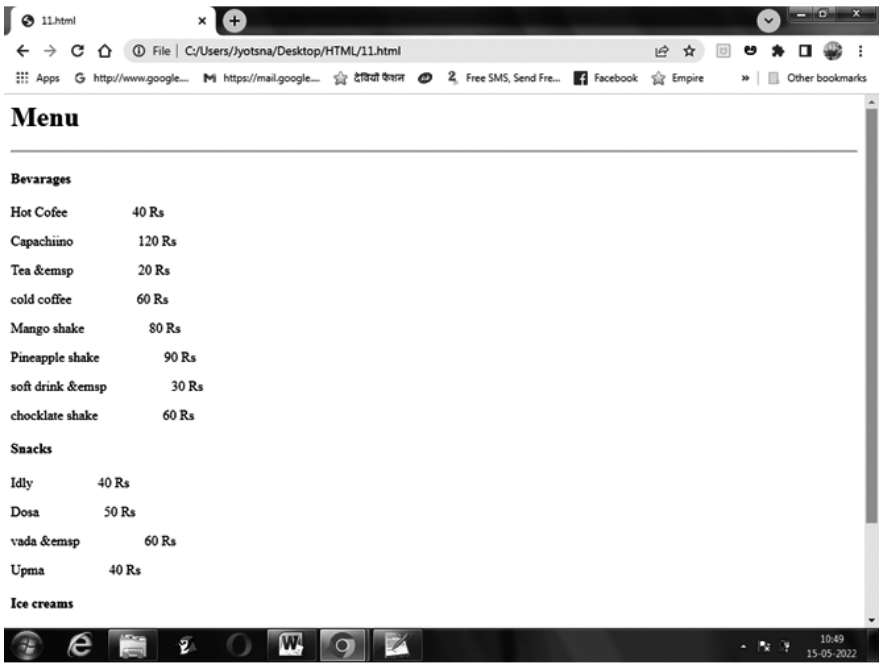
```
<body>
<div class="content_wrap">
<p>paragraph #1</p>
<p>paragraph #2 <span style="font-weight:bold">with &lt;span&gt; element embedded</span></p>
</div>
</body>
</html>
```

Q8. Write a HTML Menu page for Example cafe site.

Ans :

```
<div class="container">
<div class="row d-flex justify-content-center">
<div class="col-10 pb-5 text-center">
<h1 class="quicksand-font header text-uppercase">Menu</h1>
<hr>
</div>
</div>
<div class="row d-flex justify-content-center">
<div class="col-10 pb-5 text-center">
<h4 class="text-center quicksand-font text-uppercase">Bevarages</h4>
<p class="text-uppercase">Hot Cofee      40 Rs</p>
<p class="text-uppercase">Capachiino       120 Rs</p>
<p class="text-uppercase">Tea        20 Rs</p>
    <p class="text-uppercase"> cold coffee       60 Rs</p>
<p class="text-uppercase">Mango shake       80 Rs</p>
<p class="text-uppercase">Pineapple shake        90 Rs</p>
<p class="text-uppercase">soft drink        30 Rs</p>
    <p class="text-uppercase">chocklate shake       60 Rs</p>
```

```
</div>
</div>
<div class="row d-flex justify-content-center">
<div class="col-10 pb-5 text-center">
<h4 class="text-center quicksand-font text-uppercase">Snacks</h4>
<p class="text-uppercase">Idly        40 Rs</p>
<p class="text-uppercase">Dosa       50 Rs</p>
<p class="text-uppercase">vada       60 Rs</p>
    <p class="text-uppercase">Upma      40 Rs</p>
</div>
</div>
<div class="row d-flex justify-content-center">
<div class="col-10 pb-5 text-center">
<h4 class="text-center quicksand-font text-uppercase">Ice creams</h4>
<p class="text-uppercase">Butter scotch       120 Rs</p>
<p class="text-uppercase">Pista       120 Rs</p>
<p class="text-uppercase">Mango       120 Rs</p>
<p class="text-uppercase">Chocklate      120 Rs</p>
</div>
</div>
```



Q9. Write a HTML program using images, audios, videos.

Ans :

```
<!DOCTYPE HTML>
<html>
<body>
<audiocontrolsautoplay>
    <sourcesrc="/html5/audio.ogg"type="audio/ogg"/>
    <sourcesrc="/html5/audio.wav"type="audio/wav"/>
Your browser does not support the <audio> element.
    </audio>
    <videowidth="300"height="200"controlsautoplay>
    <sourcesrc="/html5/foo.ogg"type="video/ogg"/>
    <sourcesrc="/html5/foo.mp4"type="video/mp4"/>
Your browser does not support the <video> element.
</video>
</body>
</html>
```

Q10. Write a HTML program to create your time table.

Ans :

```
<!DOCTYPE html>
<html>
<body>
    <h1>TIME TABLE</h1>
    <table border="5" cellspacing="0" align="center">
        <!—<caption>Timetable</caption>—>
        <tr>
            <td align="center" height="50"
                width="100"><br>
            <b>Day/Period</b></br>
        </td>
        <td align="center" height="50"
            width="100">
            <b>I<br>9:30-10:20</b>
        </td>
```

```
<td align="center" height="50"
    width="100">
    <b>II<br>10:20-11:10</b>
</td>
<td align="center" height="50"
    width="100">
    <b>III<br>11:10-12:00</b>
</td>
<td align="center" height="50"
    width="100">
    <b>12:00-12:40</b>
</td>
<td align="center" height="50"
    width="100">
    <b>IV<br>12:40-1:30</b>
</td>
<td align="center" height="50"
    width="100">
    <b>V<br>1:30-2:20</b>
</td>
<td align="center" height="50"
    width="100">
    <b>VI<br>2:20-3:10</b>
</td>
<td align="center" height="50"
    width="100">
    <b>VII<br>3:10-4:00</b>
</td>
</tr>
<tr>
<td align="center" height="50">
    <b>Monday</b> </td>
<td align="center" height="50">Eng</td>
<td align="center" height="50">Mat</td>
<td align="center" height="50">Che</td>
```

```
        <td rowspan="6" align="center" height="50">
            <h2>L<br>U<br>N<br>C<br>H</h2>
        </td>
        <td colspan="3" align="center"
            height="50">LAB</td>
        <td align="center" height="50">Phy</td>
    </tr>
    <tr>
        <td align="center" height="50">
            <b>Tuesday</b>
        </td>
        <td colspan="3" align="center"
            height="50">LAB
        </td>
        <td align="center" height="50">Eng</td>
        <td align="center" height="50">Che</td>
        <td align="center" height="50">Mat</td>
        <td align="center" height="50">SPORTS</td>
    </tr>
    <tr>
        <td align="center" height="50">
            <b>Wednesday</b>
        </td>
        <td align="center" height="50">Mat</td>
        <td align="center" height="50">phy</td>
        <td align="center" height="50">Eng</td>
        <td align="center" height="50">Che</td>
        <td colspan="3" align="center"
            height="50">LIBRARY
        </td>
    </tr>
    <tr>
        <td align="center" height="50">
        <b>Thursday</b>
        </td>
```

```
                <td align="center" height="50">Phy</td>
                <td align="center" height="50">Eng</td>
                <td align="center" height="50">Che</td>
                <td colspan="3" align="center"
                    height="50">LAB
                </td>
                <td align="center" height="50">Mat</td>
            </tr>
            <tr>
                <td align="center" height="50">
                    <b>Friday</b>
                </td>
                <td colspan="3" align="center"
                    height="50">LAB
                </td>
                <td align="center" height="50">Mat</td>
                <td align="center" height="50">Che</td>
                <td align="center" height="50">Eng</td>
                <td align="center" height="50">Phy</td>
            </tr>
            <tr>
                <td align="center" height="50">
                    <b>Saturday</b>
                </td>
                <td align="center" height="50">Eng</td>
                <td align="center" height="50">Che</td>
                <td align="center" height="50">Mat</td>
                <td colspan="3" align="center"
                    height="50">SEMINAR
                </td>
                <td align="center" height="50">SPORTS</td>
            </tr>
        </table>
</body>
</html>
```

Output

TIME TABLE

Day/Period	I 9:30-10:20	II 10:20-11:10	III 11:10-12:00	12:00-12:40	IV 12:40-1:30	V 1:30-2:20	VI 2:20-3:10	VII 3:10-4:00
Monday	Eng	Mat	Che	LUNCH	LAB			Phy
Tuesday	LAB				Eng	Che	Mat	SPORTS
Wednesday	Mat	phy	Eng		Che	LIBRARY		
Thursday	Phy	Eng	Che		LAB			Mat
Friday	LAB				Mat	Che	Eng	Phy
Saturday	Eng	Che	Mat		SEMINAR			SPORTS

Q11. Write a HTML program to create a form using text inputs, password inputs, multiple line text input, buttons, check boxes, radio buttons, select boxes, file select boxes.

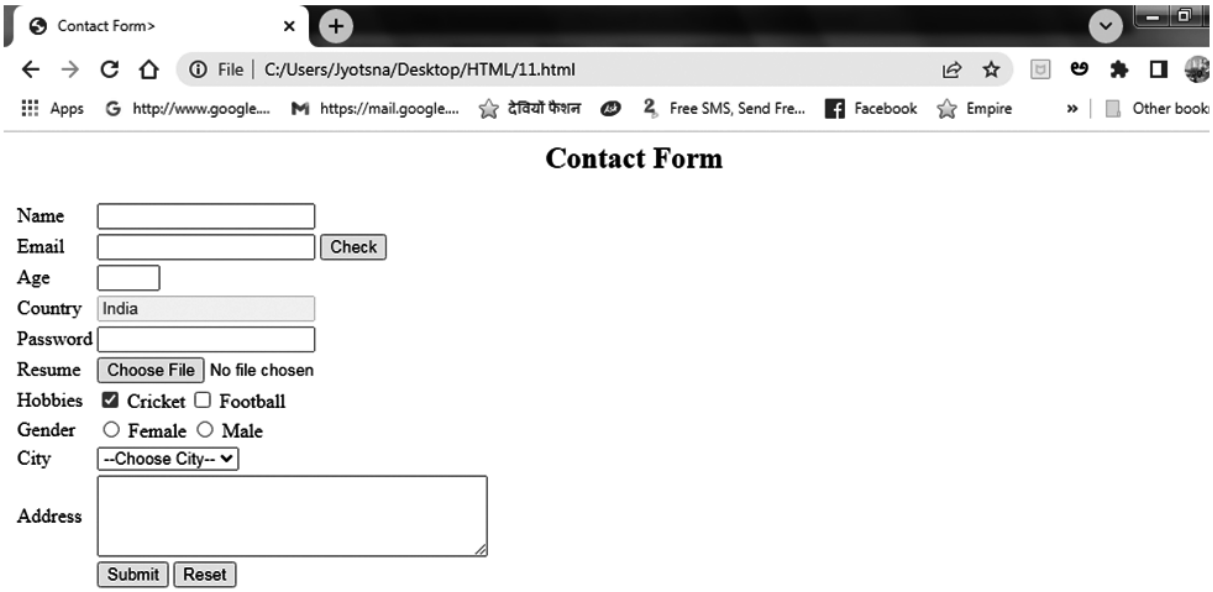
Ans :

```
<html>
<title> Contact Form></title>
<body>
<h2 align = “center”> Contact Form</h2>
<form>
<table>
<tr>
<td>
<label for=”uname”>Name</label>
</td>
<td>
<input type=”text” id=”uname” name=”username”>
</td>
</tr>
<tr>
<td>
<label for=”uemail”>Email</label>
</td>
<td>
<input type=”text” id=”uemail” name=”usermail”>
```

```
<button type="button">Check</button>
</td>
</tr>
<tr>
<td>
<label for="age">Age</label>
</td>
<td>
<input type="text" name="userage" id="age" size="2" maxlength="2">
</td>
</tr>
<tr>
<td>
<label>Country</label>
</td>
<td>
<input type="text" value="India" name="country" disabled>
</td>
</tr>
<tr>
<td>
<label for="pass">Password</label>
</td>
<td>
<input type="password" id="pass">
</td>
</tr>
<tr>
<td>
<label for="res">Resume</label>
</td>
<td>
<input type="file" id="res">
</td>
</tr>
```

```
<tr>
<td>
<label>Hobbies</label>
</td>
<td>
<label>
<input type="checkbox" checked> Cricket
</label>
<label>
<input type="checkbox"> Football
</label>
</td>
</tr>
<tr>
<td>
<label>Gender</label>
</td>
<td>
<label>
<input type="radio" value="f" name="gender"> Female</label>
<label>
<input value="m" type="radio" name="gender"> Male</label>
</td>
</tr>
<tr>
<td>
<label for="city">City</label>
</td>
<td>
<select id="city" name="city">
<option disabled selected>—Choose City—</option>
<optgroup label="Metros">
```

```
<option>New Delhi</option>
<option>Mumbai</option>
<option>Channai</option>
<option>Kolkata</option>
</optgroup>
<optgroup label="Others">
<option>Noida</option>
<option>Gurgram</option>
<option>Faridabad</option>
<option>Gaziabad</option>
</optgroup>
</select>
</td>
</tr>
<tr>
<td>
<label>Address</label>
</td>
<td>
<textarea rows="4" cols="40"> </textarea>
</td>
</tr>
<tr>
<td> </td>
<td>
<input type="submit" value="Submit">
<input type="reset">
</td>
</tr>
</table>
</form>
</body>
</html>
```



Q12. Write a HTML program to create frames and links between frames.

Ans :

Home.html

```
<html>
<head>
<title>Home</title>
</head>
<frameset rows="25%,*">
<frame src="frame1.html">
<frameset cols="25%,*">
<frame src="frame2.html" name="f2">
<frame src="frame3.html" name="f3">
</frameset>
</html>
```

Frame1.html

```
<html>
<head><title>frame1</title>
</head>
<body bgcolor="blue">
<h1 style="color.green;font-size:15pt">
<marquee bgcolor="#cccccc" loop="-1" scrollamount="6" width="100%"> THIRUVALLUVAR COLLEGE OF ENGINEERING AND TECHNOLOGY </marquee>
```

```
</h1>
</body>
</html>
```

Frame2.html

```
<html>
<head> <title>frame2</title>
<style type="text/css">
h1
{
font-size:25pt;color:pink;
}
</style>
</head>
<body bgcolor="red">
<h1>click the link</h1>
<a href="intro.html" target=f3>Introduction</a> <br>
<a href="dept.html" target=f3>Departments</a> <br>
<a href="ad.html" target=f3>ADDRESS</a> <br>
<a href="feed.html" target=f3>Feedback</a> <br>
<a href="gall.html" target=f3>Gallery</a> <br>
</body>
</html
```

Frame3.html

```
<html>
<head> <title>1st page</title>
<link rel="stylesheet" type="text/css" href="C:\Documents and
Settings\Administrator\Desktop\ab\css1.css"/>
</head>
<body bgcolor="tan">
<h2> <center>YOU ARE IN HOME PAGE</center> </h2>
</body>
</html>
```

Intro.html

```
<html>
<head> <title>intro</title>
</head>
```

<body bgcolor = "black" >

<font color = red >

Welcome to Thiruvalluvar College of Engineering and Technology -Affiliated to Anna University < br >

< br > "Thiruvalluvar College of Engineering and Technology resolves to mould a human task force useful to the society through transparent methods that lead to continuous improvement of the resources and state-of-the-art methodologies conforming to recognized standards."

</body >

</html >

Ad.html

<html >

<head > <title > ADDRESS </title >

</head >

<body bgcolor = "black" >

<font color = red >

Name:Thiruvalluvar College of Engineering and Technology < br >

Location:Vandavasi < br >

Contact No:04183-221444 < br >

Website: www.google.co.in < br >

</body >

</html >

Dept.html

<html >

<head > <title > Departments </title >

</head >

<body >

<table border = 2 >

<tr >

```
<th>Dept code</th>
<th>Dept name</th>
</tr>
<tr>
<td>01</td>
<td>CSE</td>
</tr>
<tr>
<td>02</td>
<td>ECE</td>
</tr>
<tr>
<td>03</td>
<td>EEE</td>
</tr>
<tr>
<td>04</td>
<td>IT</td>
</tr>
<tr>
<td>05</td>
<td>MECH</td>
</tr>
<tr>
<td>06</td>
<td>AERO</td>
</tr>
</table>
</div>
</body>
</html>
```

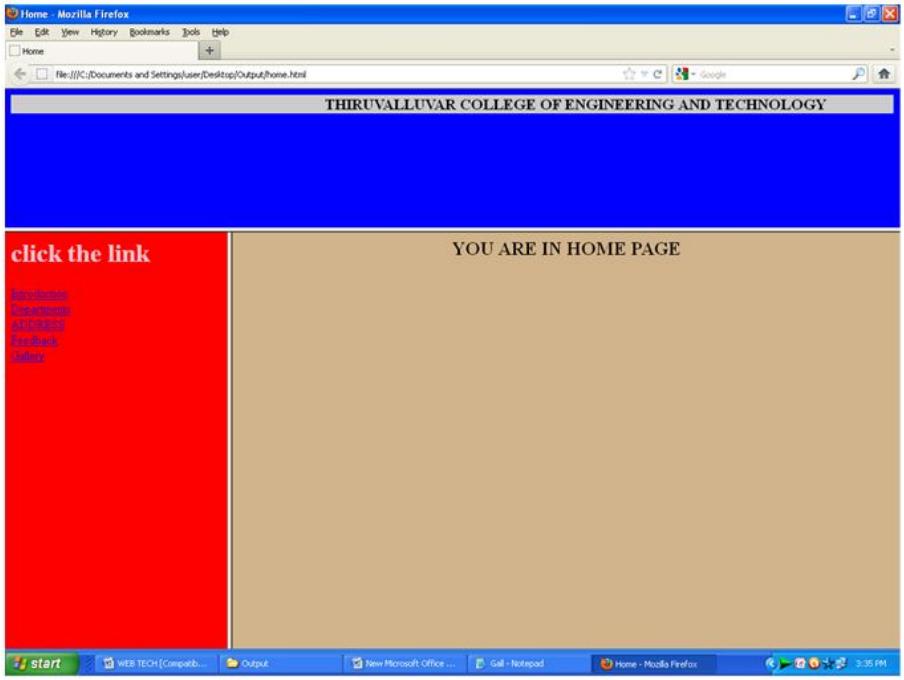
Feed.html

```
<html>
<head><title>feed</title>
</head>
```

```
<body bgcolor="black">
<p>
<font color=green>
To give your feedback mail to google_feedback@edu.in
</font>
</p>
</body>
</html>
```

Gall.html

```
<html>
<head> <title>gall</title>
</head>
<body bgcolor="pink">
<p>
<font color=blue>
College Front View</font>
</p>
<imgsrc="file:///d:/google.JPG" height="300" width="400"/>
</body>
</html>
```

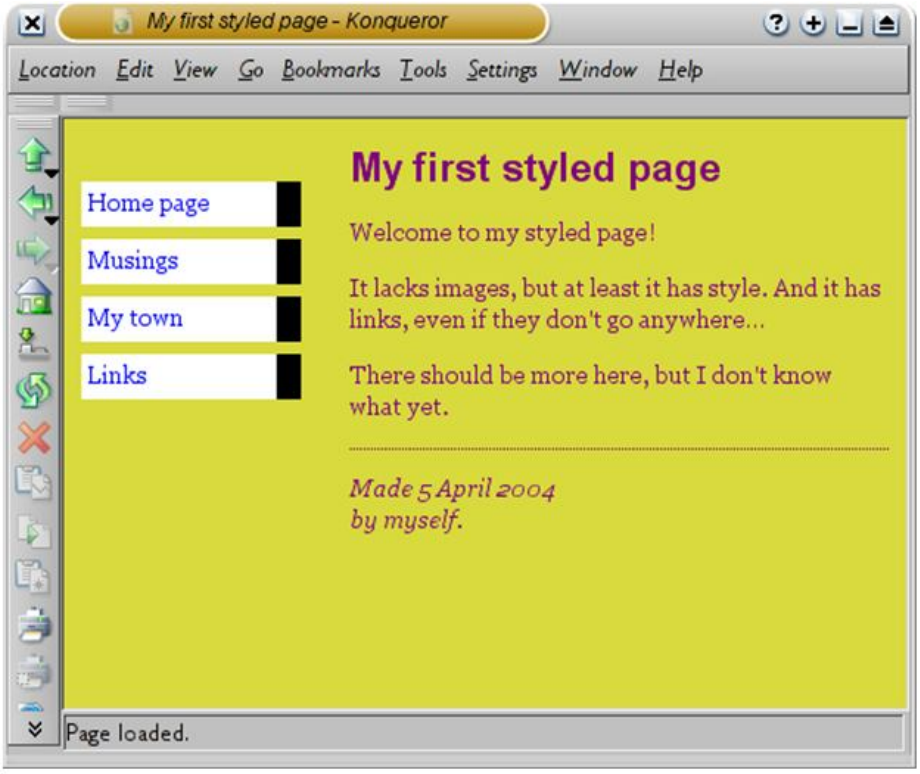


Q13. Write a HTML program to create different types of style sheets.

Ans :

```
body {
padding-left: 11em;
font-family: Georgia, "Times New Roman",
      Times, serif;
color: purple;
background-color: #d8da3d }
ul.navbar {
list-style-type: none;
padding: 0;
margin: 0;
position: absolute;
top: 2em;
left: 1em;
width: 9em }
h1 {
font-family: Helvetica, Geneva, Arial,
SunSans-Regular, sans-serif }
ul.navbar li {
background: white;
margin: 0.5em 0;
padding: 0.3em;
border-right: 1em solid black }
ul.navbar a {
text-decoration: none }
a:link {
color: blue }
      a:visited {
color: purple }
address {
margin-top: 1em;
padding-top: 1em;
border-top: thin dotted }
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>My first styled page</title>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>
```



Q14. Write a HTML program to create CSS on links, lists, tables and generated content.

Ans :

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
text-align: left;
width: 300px;
margin: 0 auto;
font-size: 1.2rem;
```

```
font-family: sans-serif;
}
p {
line-height: 1.4;
}
ul {
padding: 0;
width: 100%;
}
li {
display: inline;
}
a {
outline: none;
text-decoration: none;
display: inline-block;
width: 19.5%;
margin-right: 0.625%;
text-align: center;
line-height: 3;
color: black;
}
li:last-child a {
margin-right: 0;
}
a:link, a:visited, a:focus {
background: yellow;
}
a:hover {
background: orange;
}
a:active {
background: red;
color: white;}
h1 {
color: green;
}
table,
th,
```

```
td {
/* Styling the border. */
border: 1.5px solid blue;
}
</style>
</head>
<body>
<h1>CSS Tables</h1>
<h2>Add border to table:</h2>
    <p> Select your favorite and click
</p>
<table>
<tr><ul>
<th><li><a href="#">Home</a></li></th>
<th><li><a href="#">Pizza</a></li></th>
<th><li><a href="#">Music</a></li></th>
<th><li><a href="#">Wombats</a></li></th>
<th><li><a href="#">Finland</a></li></th>
</ul>
</tr>
</table>
</html>
```

Q15. Write a HTML program to create your college web site using multi column layouts.

Ans :

```
<!DOCTYPE html>
<html>
<head>
<title>
    Website Layout
</title>
<style>
    /* CSS property for header section */
    .header {
    background-color: green;
    padding: 15px;
    text-align: center;
    }
    /* CSS property for navigation menu */
    .nav_menu {
    overflow: hidden;
    background-color: #333;
}
.nav_menu a {
float: left;
display: block;
color: white;
text-align: center;
padding: 14px 16px;
text-decoration: none;
}
.nav_menu a:hover {
background-color: white;
color: green;
}
</style>
</head>
```

```
<body>
    <!-- header of website layout -->
    <div class = "header">
    <h2 style = "color:white;font-size:200%;">
    ABC College of Engineering and Technology
</h2>
</div>
<!-- navigation menu for website layout -->
<div class = "nav_menu">
<a href = "#">Department of CSE</a>
<a href = "#">Department of ECE</a>
<a href = "#">Department of EEE</a>
<a href = "#">Department of IT</a>
<a href = "#">Department of H&S</a>
</div><br>
<center style = "font-size:200%;">
Welcome to the Home page of ABC College of Engineering and
Technology
 </center>
</body>
</html>
```

Welcome to the Home page of ABC College of Engineering and Technology

Q16. Write a HTML program to create your college web site using for mobile device.

Ans :

```
<html>
<head>
<center>ALL STYLE SHEETS</center>
<title>USE of INTERNAL and EXTERNAL STYLESHEETS
</title>
<link rel="stylesheet"href="xyz.css" type="text/css">
<style type="text/css">
.vid{font-family:verdana;font-style:italic;color:red;text-align:center}
.ani{font-family:tahoma;font-style:italic;font-size:20;text-align:center;}
font{font-family:georgia;color:blue;font-size:20}
ul{list-style-type:circle}
</style>
</head>
<body>
<ol style="list-style-type:lower-alpha">
<b>A. GROUP OF COLLEGES</b><br><br><br>
<li>RajEngineeringCollege
<li>DrUniversityandResearchInstitute
<li>RajCollege of Engineering,Bangalore
<li>TamilnaduCollege of ArtsandScience
</ol>
<p style="font-size:20pt;color:purple">A.C.S GROUP OF COLLEGES</p>
<p class="ani">CSS Group of colleges is owned by XXX.<br>Itis approved
by
InternationalCouncil
Itisaffliated to JKK University.<br></p>
<h2 class="vid">DES College of Engg</h2>
<br>
<font>Itis an ISO certified Institution</font><br>
<br>
<font>
List of Courses offered
```

```
<ul>
<li>ComputerScienceandEngineering</li>
<li>Ece</li>
<li>mech</li>
<li>eee</li>
</ul>
</font>
Results of cse students
<table width="100%"cellspacing="2"cellpadding="2" border="5">
<tr>
<th>S.NAME</th>
<th>MARKS</th>
<th>RESULT</th>
</tr>
<tr>
<td align="center">Dinesh</td>
<td align="center">100</td>
<td align="center">pass</td>
</tr>
<tr>
<td align="center">Bala</td>
<td align="center">99</td>
<td align="center">pass</td>
</tr>
<tr>
<td align="center">Gopi</td>
<td align="center">98</td>
<td align="center">pass</td>
</tr>
</table>
</body>
</html>
```

Q17. Write a HTML program to create login form and verify username and password.

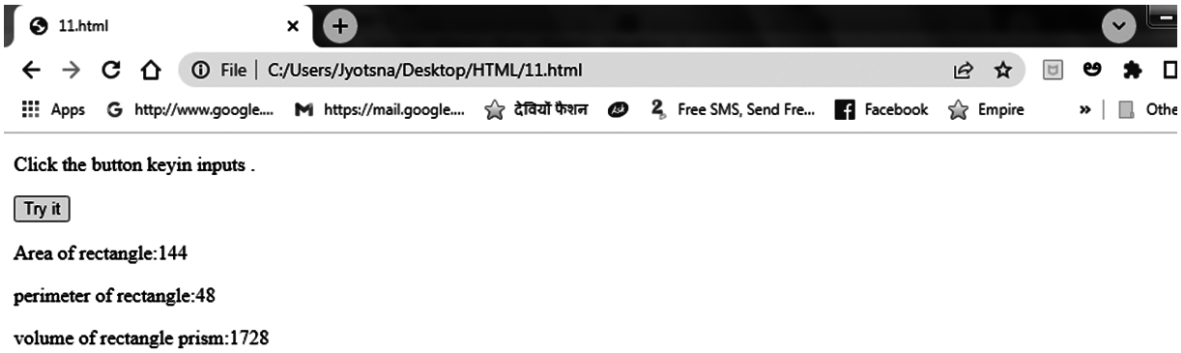
Ans :

```
<!doctype html>
<form name="loginForm" method="post" action="index.html">
<table width="20%" bgcolor="0099CC" align="center">
<tr>
<td colspan=2><center><font size=4><b>light-net login</b></font></center></td>
</tr>
<tr>
<td>Username:</td>
<td><input type="text" size=25 name="user"></td>
</tr>
<tr>
<td>Password:</td>
<td><input type="Password" size=25 name="pass"></td>
</tr>
<tr>
<td><input type="Reset"></td>
<td><input type="submit" onclick="return check(this.form)" value="Login"></td>
</tr>
</table>
</form>
<script language="javascript">
function check(form)
{
if(form.user.value == "public" &&form.pass.value == "peasants")
{
    return true;
}
else
{
    alert("Error Please Eneter Your Password or Username")
    return false;
}
}
</script>
```

Q18. Write a JavaScript program to calculate area of rectangle using function.

Ans :

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button keyininputs .</p>
<button onclick="myFunction()">Try it</button>
<p id="a"></p>
<p id="p"></p>
<p id="v"></p>
<script>
functionmyFunction() {
    var length = prompt("Enter a whole number for the length of your rectangle.");
    var width = prompt ("Enter a whole number for the width of your rectangle.");
    var depth= prompt ("Enter a whole number for the depth of your rectangle prism");
    var perimeter = (2 * length ) + (2 * width );
    var area= length * width ;
    var volume= length * width * depth;
document.getElementById("a").innerHTML =
    "Area of rectangle:" + area;
document.getElementById("p").innerHTML =
    "perimeter of rectangle:" + perimeter ;
document.getElementById("v").innerHTML =
    "volume of rectangle prism:" + volume;
}
</script>
</body>
</html>
```



Q19. Write a JavaScript program to wish good morning, good afternoon, good evening depending on the current time.

Ans :

```
<!DOCTYPE html>
<html>
<body>
<head>
<title>Show good morning good night wish as per time Javascript</title>
</head>
<script type="text/javascript">
document.write("<center><font size=+3 style='color: green;'>");
var day = new Date();
varhr = day.getHours();
if (hr>= 0&&hr<12) {
document.write("Good Morning!");
    } elseif (hr == 12) {
document.write("Good Noon!");
    } elseif (hr>= 12&&hr<= 17) {
document.write("Good Afternoon!");
    } else {
document.write("Good Evening!");
    }
document.write("</font></center>");
</script>
</html>
</body>
</html>
```

Q20. Write a JavaScript program using switch case?

Ans :

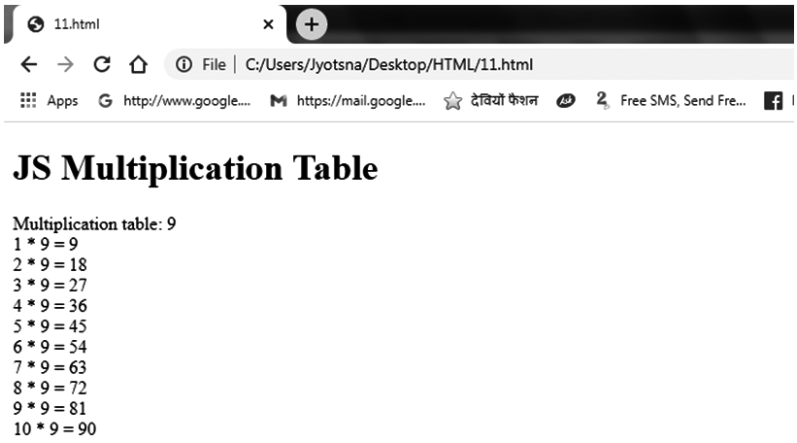
```
<!DOCTYPE html>
<html>
<body>
<head>
<title>Grade of the student</title>
</head>
<script type="text/javascript">
<script>
var grade='B';
var result;
switch(grade){
case 'A':
result="A Grade";
break;
case 'B':
result="B Grade";
break;
case 'C':
result="C Grade";
break;
default:
result="No Grade";
}
document.write(result);
</script>
</script>
</html>
</body>
</html>
```

Output: B Grade

Q21. Write a JavaScript program to print multiplication table of given number using loop.

Ans :

```
<!DOCTYPE html>
<html>
<body>
<h1>JS Multiplication Table</h1>
<script>
var table = 9;
var length = 10;
var i = 1;
document.write("Multiplication table: " + table);
for(i = 1; i <= length; i++)
document.write("<br>" + i + " * " + table + " = " + (i * table));
</script>
</body>
</html>
```



Q22. Write a JavaScript programs using any 5 events.

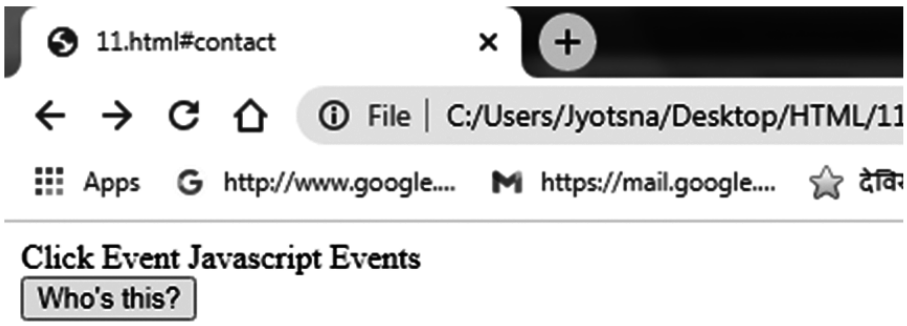
Ans :

Click Event

```
<html>
<head> Javascript Events </head>
<body>
<script language="Javascript" type="text/Javascript">
<!—
```

```
function clickevent()
{
    document.write("This is click event");
}
//—>
 </script>
 <form>
 <input type="button" onclick="clickevent()" value="Who's this?"/>
 </form>
 </body>
 </html>
```

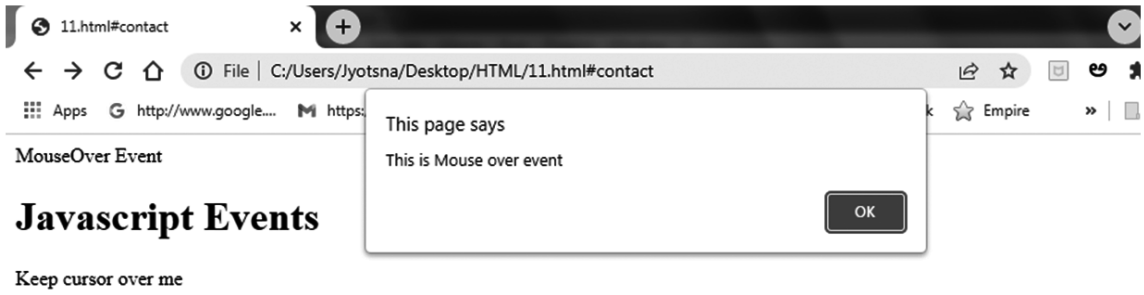
Output



MouseOver Event

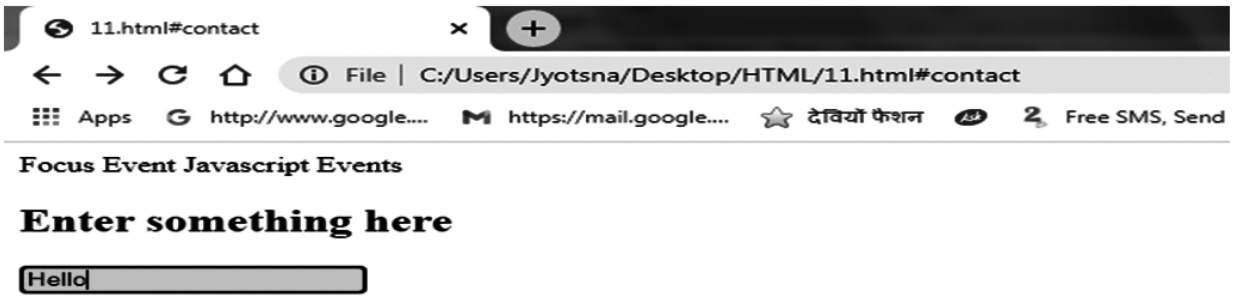
```
<html>
<head>
<h1> Javascript Events </h1>
</head>
<body>
<script language="Javascript" type="text/Javascript">
    <!—
    function mouseoverevent()
    {
        alert("This is JavaTpoint");
    }
    //—>
</script>
```

```
<p onmouseover="mouseoverevent()"> Keep cursor over me</p>
</body>
</html>
```



Focus Event

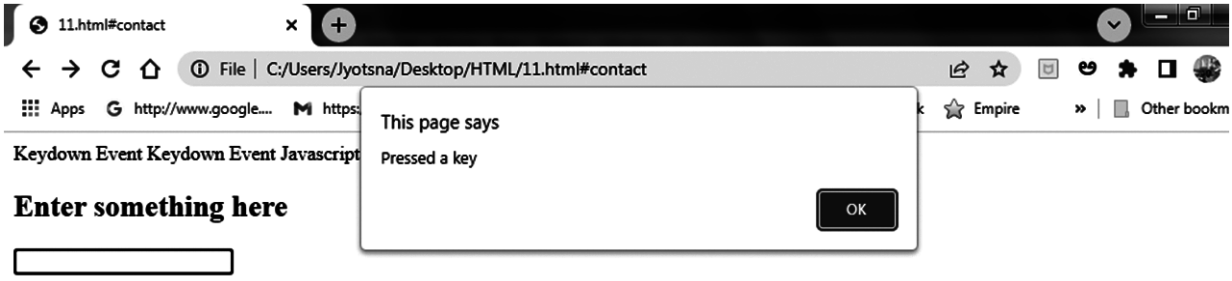
```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onfocus="focusevent()"/>
<script>
<!--
    function focusevent()
    {
        document.getElementById("input1").style.background=" aqua";
    }
//-->
</script>
</body>
</html>
```



Keydown Event

```
<html>
<head> Javascript Events</head>
```

```
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onkeydown="keydownevent()"/>
<script>
<!—
    function keydownevent()
    {
        document.getElementById("input1");
        alert("Pressed a key");
    }
//—>
</script>
</body>
</html>
```



Load event

```
<html>
<head>Javascript Events</head>
</br>
<body onload="window.alert('Page successfully loaded');">
<script>
<!—
document.write("The page is loaded successfully");
//—>
</script>
</body>
</html>
```

Q23. Write a JavaScript program using JavaScript built in objects.

Ans :

Example

Simple Program on Math Object Methods

```
<html>
    <head>
        <title>JavaScript Math Object Methods</title>
    </head>
    <body>
        <script type="text/javascript">
            var value = Math.abs(20);
            document.write("ABS Test Value : " + value +"<br>");
                var value = Math.acos(-1);
            document.write("ACOS Test Value : " + value +"<br>");
            var value = Math.asin(1);
            document.write("ASIN Test Value : " + value +"<br>");
                    var value = Math.atan(.5);
            document.write("ATAN Test Value : " + value +"<br>");
        </script>
    </body>
</html>
```

Output

ABS Test Value : 20

ACOS Test Value : 3.141592653589793

ASIN Test Value : 1.5707963267948966

ATAN Test Value : 0.4636476090008061

Example : JavaScript Date() Methods Program

```
<html>
    <body>
    <center>
        <h2>Date Methods</h2>
        <script type="text/javascript">
            var d = new Date();
            document.write("<b>Locale String:</b> " + d.toLocaleString()+"<br>");
            document.write("<b>Hours:</b> " + d.getHours()+"<br>");
            document.write("<b>Day:</b> " + d.getDay()+"<br>");
            document.write("<b>Month:</b> " + d.getMonth()+"<br>");
            document.write("<b>FullYear:</b> " + d.getFullYear()+"<br>");
            document.write("<b>Minutes:</b> " + d.getMinutes()+"<br>");
        </script>
    </center>
    </body>
</html>
```

Output:

Date Methods

Locale String: 6/1/2016 10:27:02 AM
Hours: 10
Day: 3
Month: 5
FullYear: 2016
Minutes: 27

Example : JavaScript String() Methods Program

```
<html>
    <body>
    <center>
        <script type="text/javascript">
            varstr = "CareerRide Info";
            var s = str.split();
            document.write("<b>Char At:</b> " + str.charAt(1)+"<br>");
            document.write("<b>CharCode At:</b> " + str.charCodeAt(2)+"<br>");
            document.write("<b>Index of:</b> " + str.indexOf("ide")+"<br>");
```

```
            document.write("<b>Lower Case:</b> " + str.toLowerCase()+"<br>");
            document.write("<b>Upper Case:</b> " + str.toUpperCase()+"<br>");
        </script>
      <center>
    </body>
</html>
```

Output:

Char At: a
CharCode At: 114
Index of: 7
Lower Case: careerride info
Upper Case: CAREERRIDE INFO

Q24. Write a JavaScript program to create registration Form with Validations.

Ans :

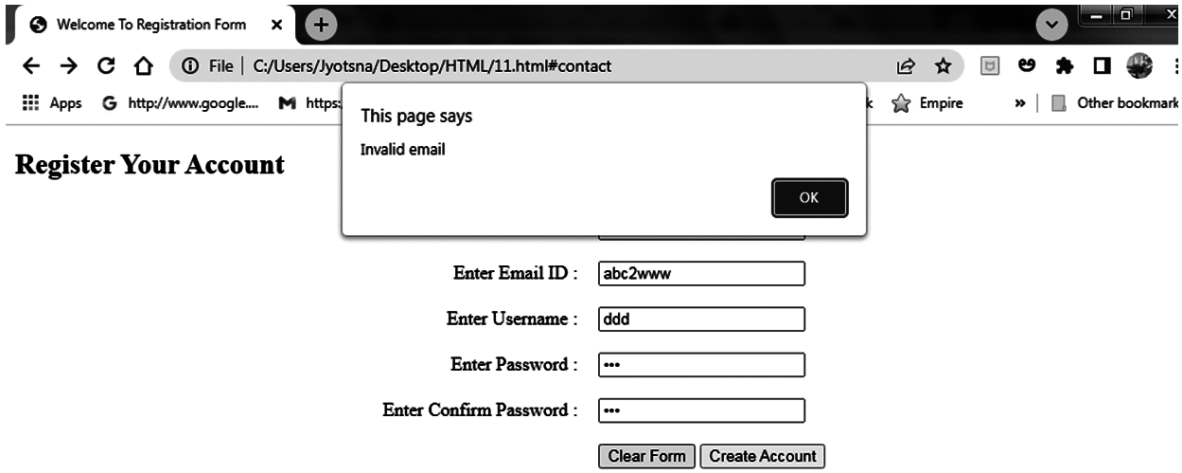
```
<!DOCTYPE html>
<html>
<head>
<title>Welcome To Registration Form</title>
<script type="text/javascript">
    function registration()
        {
        var name= document.getElementById("t1").value;
        var email= document.getElementById("t2").value;
        varuname= document.getElementById("t3").value;
        varpwd= document.getElementById("t4").value;
        varcpwd= document.getElementById("t5").value;
        //email id expression code
    varpwd_expression = /^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[#?!@$%^&*-])/;
    var letters = /^[A-Za-z]+$/;
    var filter = /^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/;
    if(name==='')
    {
    alert('Please enter your name');
}
```

```
 else if(!letters.test(name))
{
        alert('Name field required only alphabet characters');
}
        else if(email==')
{
        alert('Please enter your user email id');
}
        else if (!filter.test(email))
{
        alert('Invalid email');
}
        else if(uname==')
{
        alert('Please enter the user name.');
}
        else if(!letters.test(uname))
{
        alert('User name field required only alphabet characters');
}
        else if(pwd==')
{
        alert('Please enter Password');
}
        else if(cpwd==')
{
        alert('Enter Confirm Password');
}
        else if(!pwd_expression.test(pwd))
{
        alert ('Upper case, Lower case, Special character and Numeric letter are required in Password
        filed');
}
        else if(pwd != cpwd)
{
        alert ('Password not Matched');
}
        else if(document.getElementById("t5").value.length< 6)
{
        alert ('Password minimum length is 6');
}
```

```
        else if(document.getElementById("t5").value.length > 12)
    {
        alert ('Password max length is 12');
    }
        else
    {
        alert('Thank You for Registration & You are Redirecting to Website');
            // Redirecting to other page or webste code.
        window.location = "https://tutorial.eyehunts.com//";
     }
}
    </script>
    </head>
    <body>
    <!— Main div code —>
    <div id="main">
    <div class="h-tag">
    <h2>Register Your Account</h2>
    </div>
    <!— create account div —>
    <div class="login">
    <table cellspacing="2" align="center" cellpadding="8" border="0">
    <tr>
    <td align="right">Enter Name :</td>
    <td><input type="text" placeholder="Enter user here" id="t1" class="tb" /></td>
    </tr>
    <tr>
    <td align="right">Enter Email ID :</td>
    <td><input type="text" placeholder="Enter Email ID here" id="t2" class="tb" /></td>
    </tr>
    <tr>
    <td align="right">Enter Username :</td>
    <td><input type="text" placeholder="Enter Username here" id="t3" class="tb" /></td>
    </tr>
    <tr>
    <td align="right">Enter Password :</td>
    <td><input type="password" placeholder="Enter Password here" id="t4" class="tb" /></td>
    </tr>
    <tr>
    <td align="right">Enter Confirm Password :</td>
```

```
<td><input type="password" placeholder="Enter Password here" id="t5" class="tb" /></td>
</tr>
<tr>
<td></td>
<td>
<input type="reset" value="Clear Form" id="res" class="btn" />
<input type="submit" value="Create Account" class="btn" onclick="registration()" /></td>
</tr>
</table>
</div>
<!— create account box ending here.. —>
</div>
<!— Main div ending here... —>
</body>
</html>
```

Output



Q25. Write a XML Program to represent Student Data using DTD.

Ans :

Student.Xml

```
<?xml version="1.0"?>
<!DOCTYPE STUDENTS SYSTEM "E:\XML1\STUDENT.dtd">
<STUDENTS>
    <STUDENT>
    <STUDENTDATA>
    <NAME> SUTHA </NAME>
    <ID>  2007IT51</ID>
```

```
        <AGE> 20 </AGE>
        <ADDRESS> 12,SATHY ROAD,GOBI  </ADDRESS>
    </STUDENTDATA>
</STUDENT>
</STUDENTS>
```

Student.dtd

```
<?xml version="1.0"?>
<!ELEMENT STUDENTS (STUDENT*)>
<!ELEMENT STUDENT (STUDENTDATA*)>
<!ELEMENT STUDENTDATA (NAME,ID,AGE,ADDRESS)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT AGE (#PCDATA)>
<!ELEMENT ADDRESS (#PCDATA)>
```

Student.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"E:\XML1\STUDENT.dtd"">
<html>
<head COLOR:RED><h1 style="color:red">
<MARQUEE DIRECTION="RIGHT"><CENTER> COLLEGE OF ENGINEERING AND TECHNOLOGY </CENTER></MARQUEE></H1>
<title> STUDENT DETAILS DISPLAY </title>
</head>
<body
style="background-color:PINK"><H2 STYLE="COLOR: BLUE"> <MARQUEE> <CENTER> DEPARTMENT OF INFORMATION TECHNOLOGY </CENTER></MARQUEE><BR></H2>
<MARQUEE DIRECTION="DOWN"><H3 STYLE= "COLOR:GREEN"><CENTER>FINAL IT STUDENTS DETAILS  </CENTER></MARQUEE><BR><BR></H3>
        <CENTER><TABLE BORDER="1">
        <THEAD>
        <TR>
        <TH> NAME </TH>
        <TH> ID  </TH>
        <TH> AGE  </TH>
        <TH> ADDRESS  </TH>
        </TR>
```

```
</THEAD>
<TFOOT>
<TR>
<TH COLSPAN="4"> STUDENT CATALOG</TH>
</TFOOT>
<TR>
    <TD>SUTHA </TD>
    <TD>2007IT51</TD>
    <TD>20</TD>
    <TD>12,SATHY ROAD,GOBI</TD>
</TR>
</TABLE> </CENTER>
</body>
</html>
```

Run the html file in any browser like Internet explorer....output will be like this

Q26. Write a XML Program to represent Data using XML Schema Definition.

Ans :

employee.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.javatpoint.com"
xmlns="http://www.javatpoint.com"
elementFormDefault="qualified">
<xs:element name="employee">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="firstname" type="xs:string"/>
            <xs:element name="lastname" type="xs:string"/>
            <xs:element name="email" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

employee.xml

```
<?xml version="1.0"?>
<employee
xmlns="http://www.javatpoint.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.javatpoint.com employee.xsd">
<firstname>vimal</firstname>
<lastname>jaiswal</lastname>
<email>vimal@javatpoint.com</email>
</employee>
```

FACULTY OF SCIENCE

B.Sc. III-Year VI-Semester (CBCS) Examination

MODEL PAPER - I

WEB TECHNOLOGIES

Time : 3 Hours] [Max. Marks : 80

PART - A (8 × 4 = 32 Marks)

Note : Answer any **Eight** of the following questions.

ANSWERS

1. What is HTML? **(Unit-I, SQA - 1)**
2. Meta Elements **(Unit-I, SQA - 8)**
3. Explain about HTML <table> tag with an example **(Unit-I, SQA - 5)**
4. Explain the simple program structure of Java Script **(Unit-II, SQA - 13)**
5. What is function in javascript? **(Unit-II, SQA - 5)**
6. Declaring a Function in Java script **(Unit-II, SQA - 12)**
7. What is array in javascript? **(Unit-III, SQA - 1)**
8. Event handling in java script **(Unit-III, SQA - 2)**
9. Explain String object in Java script **(Unit-III, SQA - 9)**
10. Features of XML. **(Unit-IV, SQA - 2)**
11. XML Namespaces. **(Unit-IV, SQA - 4)**
12. What is XSL? **(Unit-IV, SQA - 8)**

PART - B (4 × 12 = 48 Marks)

Note : Answer **all** the questions.

13. (a) Write about the special characers/ character entieies in HTML. **(Unit-I, Q.No. 6)**

OR

(b) What is CSS? Explain, how CSS can be used in HTML to change the style of the document. **(Unit-I, Q.No. 15)**

14. (a) How to use prompt() dialog box in Java Script? Explain. **(Unit-II, Q.No. 3)**

OR

(b) Explain about control structures in Java Script. **(Unit-II, Q.No. 8)**

15. (a) Explain how to create and use array in Javascript. **(Unit-III, Q.No. 2)**

OR

(b) Explain String object in Java script. **(Unit-III, Q.No. 20)**

16. (a) Write the features of XML. **(Unit-IV, Q.No. 2)**

OR

(b) Explain XML DTD with an example. **(Unit-IV, Q.No. 7)**

FACULTY OF SCIENCE

B.Sc. III-Year VI-Semester (CBCS) Examination

MODEL PAPER - II

WEB TECHNOLOGIES

Time : 3 Hours] [Max. Marks : 80

PART - A (8 × 4 = 32 Marks)

Note : Answer any **Eight** of the following questions.

ANSWERS

1. usage of <img> tag **(Unit-I, SQA - 2)**
2. Write about CSS3 **(Unit-I, SQA - 11)**
3. <form> tags **(Unit-I, SQA - 6)**
4. JavaScript **(Unit-II, SQA - 1)**
5. Switch Statement **(Unit-II, SQA - 4)**
6. Scope rules in javascript **(Unit-II, SQA - 6)**
7. Javascript objects **(Unit-III, SQA - 8)**
8. What is event bubbling in java script? **(Unit-III, SQA - 6)**
9. Boolean and number object **(Unit-III, SQA - 10)**
10. What is XML schema? Explain. **(Unit-IV, SQA - 6)**
11. Explain the structure of XML data. **(Unit-IV, SQA - 3)**
12. Advantage of XSLT **(Unit-IV, SQA - 9)**

PART - B (4 × 12 = 48 Marks)

Note : Answer **all** the questions.

13. (a) Explain about HTML <table> tag with an example. **(Unit-I, Q.No. 9)**

OR

(b) Explain with an example how to inter link the document in HTML. **(Unit-I, Q.No. 13)**

14. (a) Explain about the memory allocation concept in Java Script. **(Unit-II, Q.No. 5)**

OR

(b) Explain Java script Export and Import Modules. **(Unit-II, Q.No. 15)**

15. (a) Explain JavaScript onresize event. **(Unit-III, Q.No. 17)**

OR

(b) What are cookies? How to create a Cookie in JavaScript? **(Unit-III, Q.No. 25)**

16. (a) What is XSL? Explain about it. **(Unit-IV, Q.No. 10)**

OR

(b) What is a Rich Internet Application (RIA)? **(Unit-IV, Q.No. 15)**

FACULTY OF SCIENCE

B.Sc. III-Year VI-Semester (CBCS) Examination

MODEL PAPER - III

WEB TECHNOLOGIES

Time : 3 Hours] [Max. Marks : 80

PART - A (8 × 4 = 32 Marks)

Note : Answer any **Eight** of the following questions.

ANSWERS

1. Special characers/ character entieies in HTML **(Unit-I, SQA - 3)**
2. CSS User Stylesheets **(Unit-I, SQA - 12)**
3. Define Internal linking **(Unit-I, SQA - 7)**
4. Features of Java Script **(Unit-II, SQA - 9)**
5. Counter-controlled Repetitions **(Unit-II, SQA - 10)**
6. Declaring a Function in Java script **(Unit-II, SQA - 12)**
7. Onresize event **(Unit-III, SQA - 7)**
8. Onfocus event in javascript **(Unit-III, SQA - 4)**
9. Document Object Model **(Unit-III, SQA - 11)**
10. XML **(Unit-IV, SQA - 1)**
11. XML vocabularies **(Unit-IV, SQA - 7)**
12. What is XML DOM? **(Unit-IV, SQA - 10)**

PART - B (4 × 12 = 48 Marks)

Note : Answer **all** the questions.

13. (a) Explain CSS User Stylesheets. **(Unit-I, Q.No. 25)**

OR

(b) Explain briefly about <form> tags. **(Unit-I, Q.No. 11)**

14. (a) Explain various arithemetic operators used in JavaScript with examples. **(Unit-II, Q.No. 6)**

OR

(b) What is function in javascript? Explain about user defined functions. **(Unit-II, Q.No. 16)**

15. (a) Explain about Javascript objects with example. **(Unit-III, Q.No. 18)**

OR

(b) Write about Document Object Model. **(Unit-III, Q.No. 23)**

16. (a) How AXAX can be communicated with XML file? Explain. **(Unit-IV, Q.No. 17)**

OR

(b) Differentiate between traditional web applications and ajax applications. **(Unit-IV, Q.No. 14)**