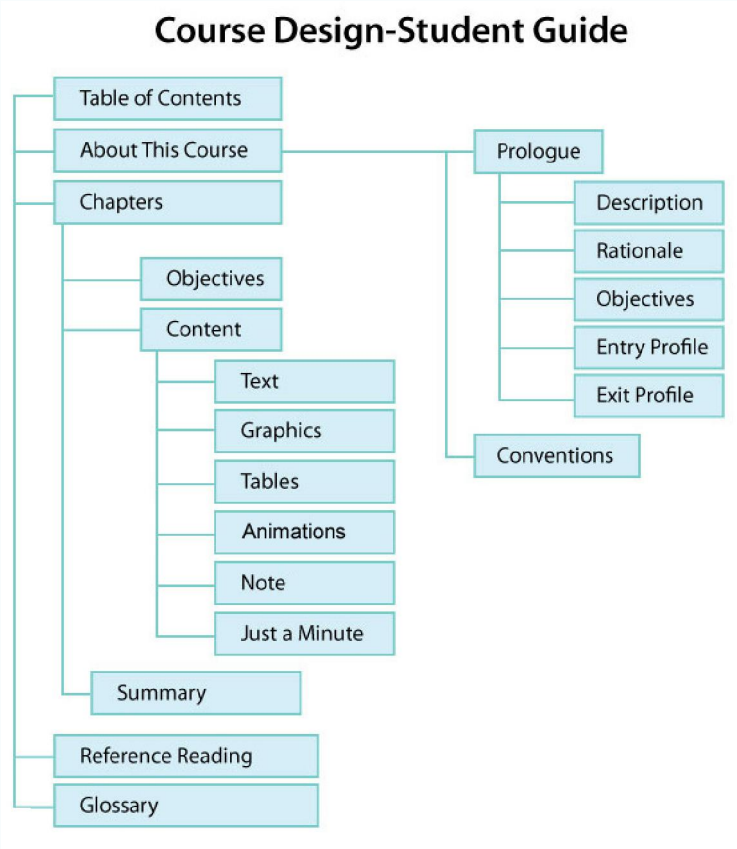

HOME
PROFILE
VIDEO
FAVOURITE
BOOKMARK
Student Guide
LEARN MORE
HTML5
Programming
HTML
5
LEARN MORE
LEARN MORE
LEARN MORE
LATEST NEWS
MEMBER
CONTRIBUTOR
MEMBER LOGIN
NIIT
1800 765 4321


Course Design-Student Guide
Table of Contents
About This Course
Chapters
Objectives
Content
Text
Graphics
Tables
Animations
Note
Just a Minute
Summary
Reference Reading
Glossary
Prologue
Description
Rationale
Objectives
Entry Profile
Exit Profile
Conventions

Trademark Acknowledgements

All products are registered trademarks of their respective organizations.
All software is used for educational purposes only.

HTML5 Programming_SG /13-M03-V01

About This Course

Prologue

Description

The course, HTML5 Programming, provides an introduction to the Hypertext Markup Language (HTML). The course covers the need for HTML. It explains the various types of HTML tags and elements used to create Web pages. In addition, the course covers Cascading Style Sheet (CSS), which is used to apply formatting styles on HTML elements to enhance the look and feel of Web pages. The course also discusses how to structure data on a Web page by using tables and frames. Moreover, it discusses how to create dynamic Web pages by using JavaScript and HTML forms. In addition, the course introduces the canvas element used to create shapes and games. It also introduces the use of jQuery to add visual effects to Web pages. Finally, it discusses how to locate a users' location and store data offline.

Rationale

Website is a representation of your personal and business reflection. Today, websites have become a vital need of every organization, enabling them to reach a wide range of audience to showcase their products and services, therefore, allowing them to stay ahead of the competitors. Services through websites are available 24*7.

Technically, it is a collection of related Web pages containing content such as text, images, audio, and video. Web pages are built using markup languages such as HTML & XML. Client-side scripting languages such as JavaScript add dynamism to these Web pages making them interactive and intuitive.

Objectives

After completing this course, the students will be able to:

- Introduce HTML
- Create an HTML Web page
- Work with styles
- Apply transitions, animations, and transformations
- Create tables
- Access multiple Web pages by using frames
- Understand scriptings
- Implement JavaScript in Web pages
- Use variables, operators, and control structures
- Implement functions
- Design an HTML form
- Manipulate the components of a Web page
- Introduce canvas
- Transform and animate canvas elements
- Explore jQuery
- Add visual effects using jQuery
- Implement image rollover
- Create image gallery
- Implement geolocation
- Implement offline support

Entry Profile

The students who want to undergo this course are recommended to have the knowledge of programming logic and techniques.

Exit Profile

After completing this course, the students will be able to create and design Web pages using HTML.

Conventions

Convention	***Indicates...***
	*Note*
	*Animation*
	*Just a Minute*

Chapter 1

Getting Started with HTML

In the present day scenario, the Internet is one of the most popular medium to accomplish various tasks, such as accessing information, communicating through emails, socializing with the help of networking sites, shopping, or booking online tickets. There are several popular websites that help us accomplish the preceding tasks.

These websites consist of one or more related Web pages. These Web pages could be either static or dynamic. One of the most popular languages for designing a Web page is *Hyper Text Markup Language (HTML)*.

This chapter introduces you to HTML and discusses how to create an HTML Web page.

Objectives

In this chapter, you will learn to:

- Introduce HTML
- Create an HTML Web page

Introducing HTML

HTML is a markup language that enables you to create attractive and interactive websites. A markup language provides a way to describe the structure of a Web page, specifying how text or graphics are displayed on the Web page. It enables you to present the text in a readable format. For example, on the front page of the newspapers, certain news headings are bigger than the others. The news headings are followed by the details of the news stories in paragraphs, sometimes pointing to a different page. If the details are on a different page, a page number is specified along with the story or the heading. This page number is similar to the markup text. The following figure shows how text is presented in different styles in a newspaper.

The Text Displayed in Different Styles in a Newspaper

The way text is presented in a newspaper is similar to the way the markup language defines, processes, and presents the content. The markup language specifies the code for defining, processing, and presenting the text on Web pages. These codes are called tags.

Web pages are either static or dynamic. The static Web pages are delivered to the users exactly as these are stored. The content of these Web pages is not updated dynamically. In contrast, in dynamic Web pages, the content is rendered at the time of request. The user gets the latest and most recently-updated information in case of dynamic Web pages. For example, while surfing any website on the Internet, you must have seen the Contact Us page that shows how to reach or contact the various offices of an organization. The content of this page is delivered to the users without any change. This is known as a static Web page. Now, consider an example of a dynamic Web page. If you want to access your Gmail account, you need to enter your credentials. Depending on whether the credentials are correct or not, the corresponding page is displayed. For example, if your credentials are valid, your inbox page is displayed. If your credentials are not valid, a page displaying a message that the credentials provided are incorrect appears to you. This is a dynamic behavior of a Web page.

These Web pages or websites can be accessed through the Internet. Among all technologies, the Internet has been the fastest growing technology. With the Internet, organizations find a medium through which they could reach a larger range of people irrespective of their geographical locations. The Internet enables organizations to share and access information on the Internet and corporate Intranet. This information can be accessed anywhere, anytime. This information is stored in a storehouse called *World Wide Web (WWW)*. The information can be accessed by using Web pages.

The Internet is an interconnected network of computers across the globe. Today, it has proliferated into every sphere of life, such as schools, banks, and hospitals. It helps in performing various activities, such as booking tickets for travelling and movies, listening to music, watching movies, playing games, and searching information. Over the years, the Internet has grown tremendously and is helping in the following areas:

- **Access to information**: The Internet provides access to information on various topics, such as sports, news, and education.
- **Communication**: The Internet enables easy and rapid communication across the world through a number of services, such as emails, chats, and social networks.
- **Data transfer**: The Internet helps in transferring a large amount of data, such as files and images.
- **Electronic-commerce (e-commerce)**: The Internet is also used as a medium of trade by buyers and sellers, increasing their reach and reducing time and geographical gaps. This is known as e-commerce, which is defined as the process of buying and selling of goods and services electronically.

The Web or WWW is a collection of resources on varied topics that can be accessed through the Internet. These resources are stored in the form of Web pages and may contain text, graphics, audio, and video content.

WWW is a collection of Web pages that are scattered but interlinked. This interconnection among Web pages of a website is achieved by using a hyperlink. A hyperlink is the highlighted or underlined text on a Web page that contains the address of the linked Web page. When you click on a hyperlink, a Web page is opened, which the hyperlink is pointing to. Moving from one Web page to the other by using hyperlinks is called navigation.

Each website and every Web page has a unique address on the Internet. This address is known as the Uniform Resource Locator (URL). To view a website, an application known as a Web browser is required. It lets the users specify the URL of the website to be viewed. This URL is entered in the address bar of the browser. If the URL does not point to any particular Web page, the browser opens the Home page of the website. The Home page of a website is the first page that contains links for all the other pages of the website.

Creating an HTML Web Page

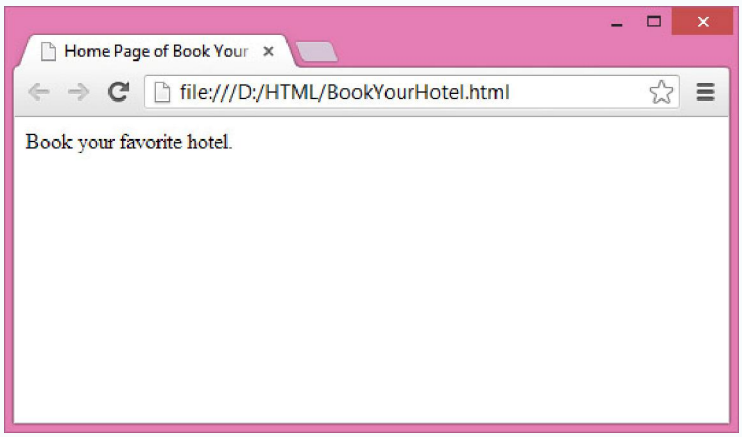
HTML is a versatile markup language that can be used on different Web browsers to publish information as a Web page. HTML provides tags that help in creating Web pages. The look and feel of these pages can further be enhanced by inserting graphics and specifying content in different font style, color, and size. This enhances the visual appeal of the Web page. HTML also allows the creation of hyperlinks to connect the different Web pages or sections of a single Web page.

Consider the following code snippet for a basic HTML document:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<TITLE> Home Page of Book Your Hotel
</TITLE>
</HEAD>
<BODY>
Book your favorite hotel.
</BODY>
</HTML>
```

The browser interprets the HTML tags to display the Web page. A Web page is viewed by using an application known as a Web browser. Most commonly-used Web browsers are Microsoft Internet Explorer and Google Chrome.

The output of the preceding code snippet is displayed in the browser, as shown in the following figure.



The Output Displayed in the Browser

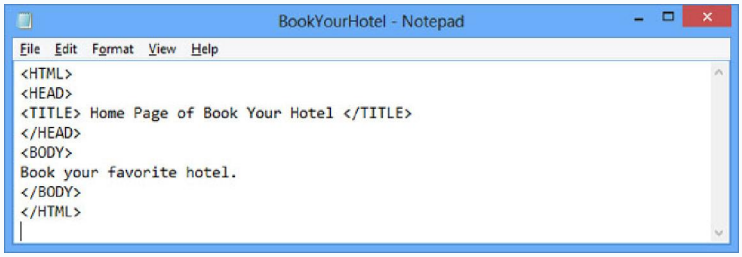
Web pages are created by using software applications known as editors. HTML editors are broadly classified into the following categories:

- Text editor
- Graphic editor

Text Editor

A text editor is an application in which the HTML code is written for creating a Web page. Notepad and EditPlus are examples of text editor applications. After writing the HTML code in the text editor, the user needs to save the file with the .htm or .html extension.

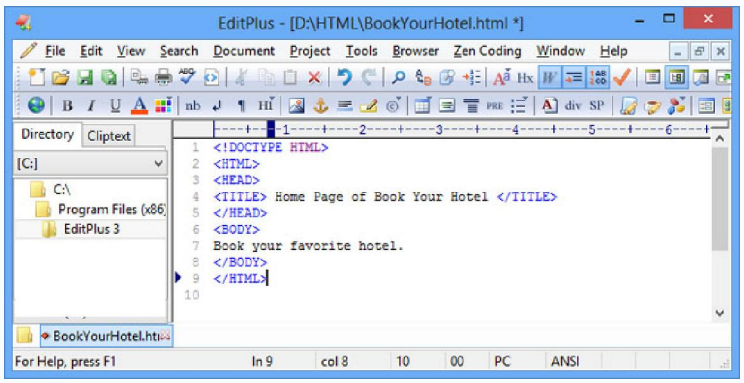
The HTML code written in Notepad is displayed, as shown in the following figure.



The HTML Code Written in Notepad

There are advanced text editors, such as EditPlus, which provide several functionalities, such as format check, line numbers, and color coding. These functionalities make writing the HTML code easier as compared to simple HTML editors, such as Notepad.

Moreover, the HTML code written in advanced HTML editors has better readability as the tags, text, and attributes are displayed in different colors. In addition, the lines of code are numbered, which further enhances readability. The HTML code written in EditPlus is displayed, as shown in the following figure.

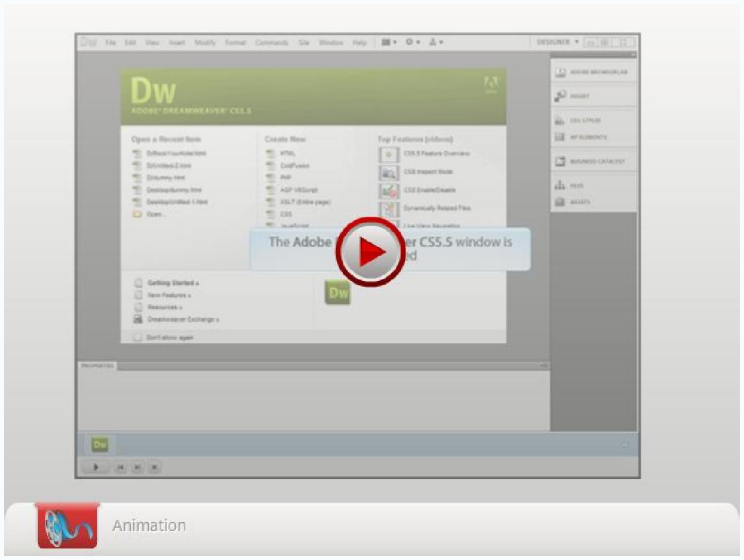


The HTML Code Written in EditPlus

Graphic Editor

Graphic editors enable a programmer to embed an HTML element into a Web page by using the drag-and-drop functionality, instead of writing the entire HTML code. This implies that it allows programmers to add an element to a Web page by simply clicking or double-clicking the corresponding icon in the toolbox. One of the extensively-used graphic editors is Dreamweaver.

Dreamweaver is a What You See Is What You Get (WYSIWYG) editor for creating Web pages. It is developed by Adobe to provide an intuitive visual interface for creating Web pages.



Identifying the Basic Structure of an HTML Page

An HTML document is created by using various tags, such as `<!DOCTYPE HTML>`, `<HTML>`, `<HEAD>`, and `<BODY>` and their attributes. Tags are special markup codes enclosed in angular brackets that define the structure of an HTML document. Some of the HTML tags, such as `<HEAD>` and `<BODY>`, are called structural tags as they create the structure of a document and do not affect the appearance of the Web page. Every tag has some attributes associated with it. These attributes modify the appearance or enhance the functionality of the tag and are provided as name-value pairs.

An HTML page contains the following structural tags:

- `<!DOCTYPE HTML>`
- `<HTML>`
- `<HEAD>`
- `<BODY>`

<!DOCTYPE HTML>

The `<!DOCTYPE>` tag provides an instruction to the browser about the version of HTML. It should be the first tag in an HTML document. It does not have any end tag, as shown in the following code snippet:

```
<!DOCTYPE HTML>
```

<HTML>

The `<HTML>` tag specifies to the browser that a document is an HTML document. The opening HTML tag, `<HTML>`, and the closing HTML tag, `</HTML>`, mark the beginning and end of a Web page. The `<HTML>` tag is a container tag that encloses the entire content of the Web page. The following code snippet is used to specify the `<HTML>` tag:

```
<HTML>
.....content of the Web page.....
</HTML>
```

A container tag is the one that needs to be closed if opened. It is, therefore, also called a paired tag and encloses the entire content inside it. Whereas, an empty tag need not be closed. It has no content inside it but possesses only the attributes.

<HEAD>

The `<HEAD>` tag is contained within the `<HTML>` and `</HTML>` tags. It is used to describe the header of the HTML document. The header is the first section of an HTML document where the global settings for the document, such as the Web page title, file format, and the last modified date, can be defined.

The following code snippet is used to specify the `<HEAD>` tag:

```
<HEAD>
...header content
</HEAD>
```

<BODY>

The <BODY> tag sets the boundary for the content in the HTML document. When the Web page is displayed in a browser, users can see the content enclosed within the opening <BODY> and closing </BODY> tags. The HTML content that can be placed within the <BODY> tag includes text and graphics to be displayed on the Web page.

The following code snippet is used to specify the <BODY> tag:

```
<BODY>
...content
</BODY>
```

Exploring the <HEAD> Tag

The head section contains the <HEAD> tag along with some other complex tags. The following tags are used in the <HEAD> section of an HTML document:

- <TITLE>
- <META>
- <BASE>
- <STYLE>
- <LINK>
- <SCRIPT>
- <NOSCRIPT>

<TITLE>

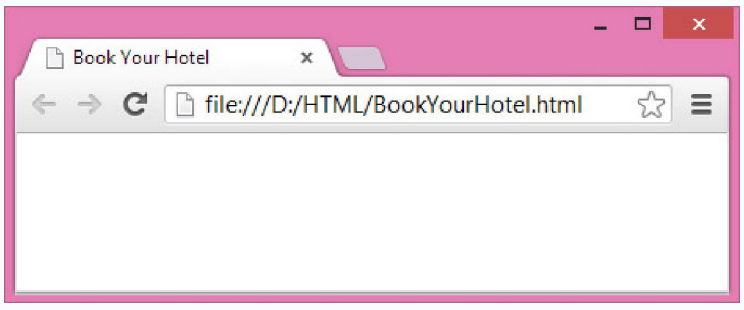
The <TITLE> tag defines the title of the document that appears in the title bar of the browser window. In addition, this title:

- Displays a title for the page to be used in the search-engine results.
- Provides a title for the page that will be added to favorites.

An HTML document can have only one title. If a title is not provided, most Web browsers display the name of the HTML document as the default title of the Web page. Consider the following code snippet for defining the <TITLE> tag:

```
<HEAD>
<TITLE> Book Your Hotel</TITLE>
</HEAD>
```

The output of the preceding code snippet is displayed, as shown in the following figure.



The Output Displaying the Usage of the <TITLE> Tag

<META>

Consider a situation where you need to specify some keywords for your website so that they can be easily searched by search engines. To implement such functionality, you can use the <META> tag. The <META> tag provides additional information about the current document in the form of name and value pairs, such as the expiry date, author name, and list of keywords. Consider the following code snippet for defining the <META> tag:

```
<HEAD>
<TITLE>Book Your Hotel</TITLE>
<META name="description"
content="This website provides you
the benefit of booking rooms in the
best hotels of US">
<META name="keywords" content="hotel,
online, booking ">
<META name="author" content="Harry
Anem">
<META http-equiv="refresh"
content="30">
</HEAD>
<BODY>
</BODY>
```

In the preceding code snippet, the <META> tag has been used with the attributes, `name`, `content`, and `http-equiv`. The `name` attribute specifies the name of the metadata. The `content` attribute is used to give the value associated with the `http-equiv` or `name` attribute. The following list describes the usage of the values that are assigned to the `name` attribute in the preceding code:

- `author`: Specifies the name of the author who has created the document.
- `description`: Specifies a short summary of the content of the Web page.
- `keywords`: Specifies the keywords used in the Web page. The `description` and `keywords` attributes are useful for the search engines to list the reference to your website when relevant keywords are entered by a user for searching.

The names of the author, summary of the contents, or the keywords used in the Web page are actually specified by using the `content` attribute of the <META> tag.

Similarly, the `http-equiv` attribute gets the information used to bind the value provided in the `content` attribute to an HTTP response header. The value, `refresh`, of the `http-equiv` attribute specifies that the page needs to be refreshed, but the time interval for the page to refresh itself is specified by using the `content` attribute. Therefore, the document will be refreshed after every 30 seconds as `30` has been set as the value to the `content` attribute.

<BASE>

The <BASE> tag specifies a base URL or address for all the related links on a page. For example, all the images that are to be used in a website during its construction and hosting are located in a particular directory. To

access these images, you can specify the absolute path of the directory by using the `href` attribute of the <BASE> tag. Thereafter, you can refer to any image in the body section of the Web page simply by specifying the image name. You can create one <BASE> tag in a document.

Consider the following code snippet for defining the <BASE> tag:

```
<!DOCTYPE HTML>
<HTML>
 <HEAD>
 <TITLE> The BASE tag Example</TITLE>
 <BASE href="d:/Images/">
 </HEAD>
 <BODY>
 <IMG src="img1.jpg">
 <BR>
 Best Hotels
 </BODY>
</HTML>
```

<STYLE>

The content of a Web page is rendered without colors and styles in the browser. This makes it non appealing to the viewers. To render the content of the Web page aesthetically and to make it attractive, you need to stylize the text. For this, you can use the <STYLE> tag.

The <STYLE> tag defines the style information associated with the HTML document. Using this element, you can specify the way the HTML elements should render in a browser. The <STYLE> tag provides various attributes, which are listed in the following table.

Attribute	*Value*	*Description*
media	*handheld screen projection tv*	*Specifies that the CSS styles are defined for which device or media.*
type	*text/css*	*Specifies the type of the style sheet. The value text/css specifies that the content in the style sheet is CSS.*

The Attributes of the <STYLE> Tag

Consider the following code snippet for defining the <STYLE> tag:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE type="text/css">
h1 {color:red;}
p {font-style:italic;}
</STYLE>
</HEAD>
<BODY>
<H1>BookYourHotel</H1>
<P>It provides online hotel booking
facility.</P>
</BODY>
</HTML>
```

In the preceding code snippet, the <STYLE> tag specifies that the heading is displayed in red color and the paragraphs in italic. When the browser comes across relevant tags in the body section, it applies the style associated with it while rendering the Web page.

The output of the preceding code snippet is displayed, as shown in the following figure.

The Output Displaying the Usage of the <STYLE> Tag

Applying styles on Web pages will be discussed in detail in the next chapter.

<LINK>

Consider a situation where you need to apply styles on your HTML content. For this, you have created style sheets. A style sheet contains the styles that need to be applied on the HTML elements to enhance their look and feel. Now, you need to link this style sheet to your HTML document to which you want to apply styles. For this, you can use the <LINK> tag.

The <LINK> tag is used to establish the relationship of the current document with other documents in a website. It can also be used to specify the external resources, such as style sheets used in the current document. It does not have any visual appearance or effects associated with it.

For example, the following code snippet can be used to link the style sheet named StyleHome.css with an HTML file named home.html:

```
<LINK href="StyleHome.css"
rel="stylesheet"/>
```

In the preceding code snippet, the `rel` attribute is used to establish a link from the current document to the external document, `StyleHome.css`. The `href` attribute specifies the name or the URL of the linked document.

In addition to the `rel` and `href` attributes, the <LINK> tag provides various other attributes. The following table describes the attributes of the <LINK> tag.

Attribute	*Value*	*Description*
hreflang	*language_code*	*Specifies the language of the text used in the linked document by using a two-letter language code, such as en for English.*

media	projection screen tv handheld	Specifies the device on which the linked document will be displayed.

The Attributes of the <LINK> Tag

<SCRIPT>

The <SCRIPT> tag specifies the client-side script, such as JavaScript, associated with the document. A script is a block of code used to add interactivity to Web pages. This tag contains various attributes that help you add scripts in a Web page. The following table describes the attributes of the <SCRIPT> tag.

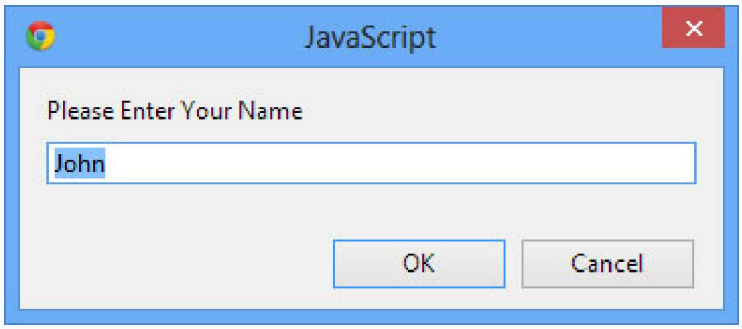
Attribute	Value	Description
async	async	Specifies that the script is executed asynchronously.
charset	UTF-8	Specifies the character encoding for an external script file.
defer	defer	Specifies that the script will be executed after a Web page has finished parsing.
src	URL	Specifies the URL of an external script file, such as d:/example.js.
type	text/javascript text/vbscript	Specifies the type of a script.

The Attributes of the <SCRIPT> Tag

Consider the following code snippet for defining the <SCRIPT> tag:

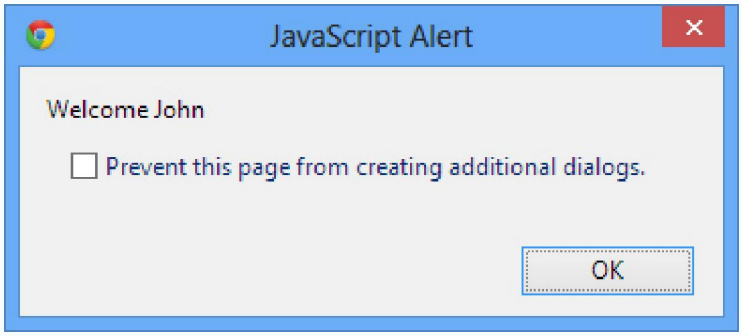
```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
var name=prompt("Please Enter Your
Name","John");
alert(name);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code snippet prompts users to enter their name by using the `prompt()` function and displays this name in a message box by using the `alert()` function. The `prompt()` function contains two parameters. The first parameter, **Please Enter Your Name**, asks the users to provide their name and the second parameter specifies **John** as the default name, as shown in the following figure.



The Prompt

The `alert()` function in the preceding code snippet displays the message, **Welcome John**, when the OK button in the prompt is clicked, as shown in the following figure.



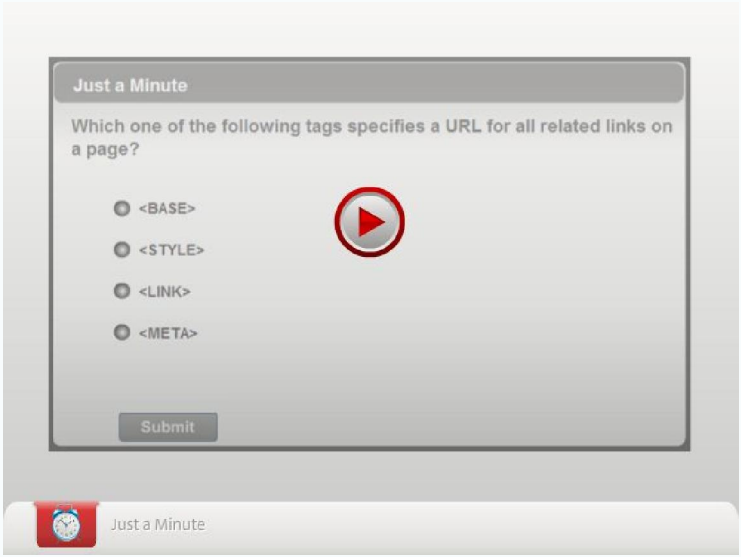
The JavaScript Alert

<NOSCRIPT>

The <NOSCRIPT> tag displays an alternate content on the browsers on which the scripts have been disabled or on the browsers that do not support client-side scripting. Consider the following code snippet for defining the <NOSCRIPT> tag:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
document.write("Welcome to
BookYourHotel website!")
</SCRIPT>
<NOSCRIPT> Your browser does not
support scripts </NOSCRIPT>
</BODY>
</HTML>
```

In the preceding code snippet, the <NOSCRIPT> tag displays the text, Your browser does not support scripts, if the scripts are not supported by the browser.



Exploring the <BODY>Tag

The body section is the central part of the HTML document required to display content on the Web page. The body section begins with the <BODY> tag. It contains the actual visible content of an HTML document. You can enhance the body of the Web page by defining headings, images, and videos.

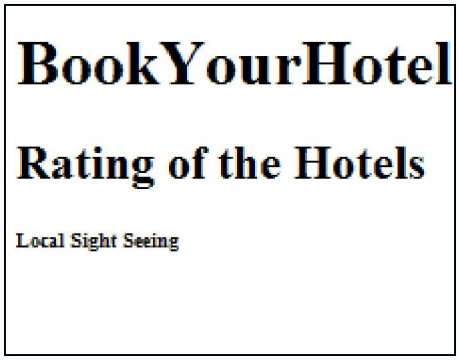
Applying Headings

With HTML, you can also define headings for a Web page. HTML allows six levels of headings, ranging from <H1> to <H6>. The <H1> tag is used to define the topmost heading, whereas the <H6> tag is used to define the lowest-level heading. The heading size indicates the heading level with <H1> being the biggest and <H6> being the smallest. Headings are used to specify the main idea of the content that follows it on the Web page.

Consider the following code for specifying heading on the BookYourHotel.com website:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
</HEAD>
<BODY>
<H1> BookYourHotel </H1>
<H2>Rating of the Hotels</H2>
<H6>Local Sight Seeing</H6>
</BODY>
</HTML>
```

In the preceding code, different heading tags are used for specifying different level headings. The output of the preceding code is displayed, as shown in the following figure.



The Output Displaying the Usage of the Heading Tags

Adding Paragraphs

You can use the following tags to add new paragraphs in an HTML document:

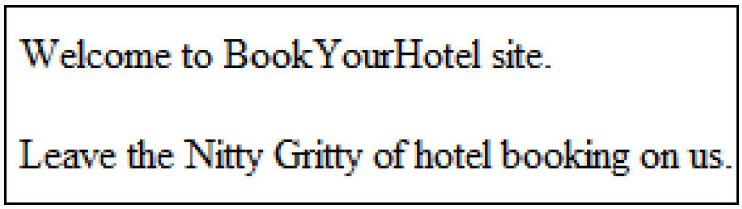
- <P>
- <DETAILS>
- <SUMMARY>

<P>

The <P> tag adds a new paragraph in the HTML document. It specifies that there is a break in the content to begin a new paragraph. The <P> tag is a container tag. Browsers automatically add one line space before and after the content specified within this tag. Consider the following code snippet for defining the <P> tag:

```
<BODY>
<P> Welcome to BookYourHotel site. </P>
<P> Leave the Nitty Gritty of hotel booking on us. </P>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.



The Output Displaying the Usage of the <P> Tag

<DETAILS>

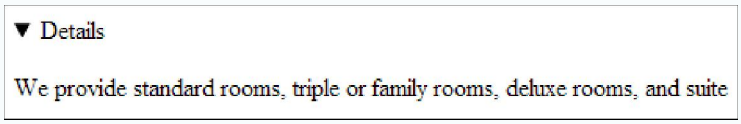
The <DETAILS> tag specifies the additional content that a user can view or hide as per requirement. It makes an interactive widget on a website that a user can open and close. The <DETAILS> tag provides the open attribute. By default, this attribute expands the content present in the <DETAILS> tag.

Consider the following code snippet for defining the <DETAILS> tag:

```
<BODY>
<DETAILS>
<P>We provide standard rooms, triple or family rooms, deluxe rooms, and suite</P>
</DETAILS>
</BODY>
```

The output of the preceding code snippet is displayed, as

shown in the following figure.



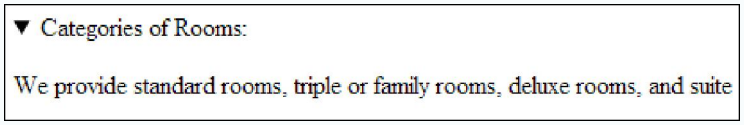
The Output Displaying the Usage of the <DETAILS> Tag

<SUMMARY>

The `<SUMMARY>` tag specifies a heading for the `<DETAILS>` tag. Users can click this heading to view or hide the details. It should be the first child element of the `<DETAILS>` tag. Consider the following code snippet for defining the `<SUMMARY>` tag:

```
<BODY>
<DETAILS>
<SUMMARY>Categories of Rooms:</
SUMMARY>
<P>We provide standard rooms, triple
or family rooms, deluxe rooms, and
suite</P>
</DETAILS>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.



The Output Displaying the Usage of the <SUMMARY> Tag

Defining a Layout

The layout of a Web page specifies how various elements or content should be displayed on the Web page. You need to define a layout of a Web page properly so that the Web page appears appealing to users. To define a layout, you can use the following tags:

- `<DIV>`
- `<SPAN>`

<DIV>

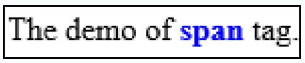
The `<DIV>` tag is used to define a section in a document. It is used to group large sections as blocks and apply different styles on these blocks. For example, you can group headings in one `<DIV>` tag and paragraph content in another `<DIV>` tag and apply different layout styles on them. This makes the Web page look more structured and attractive.

<SPAN>

The `<SPAN>` tag enables you to group and apply styles to inline elements. The `<SPAN>` tag is dependent on the `style` attribute for applying styles on the text that is enclosed inside it. `style` is a global attribute that specifies an inline style for an element. Consider the following code snippet that defines the `<SPAN>` tag:

```
<BODY>
<P>The demo of <span
style="color:blue;font-
weight:bold">span</SPAN> tag.</P>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.



The Output Displaying the Usage of the <SPAN> Tag

Formatting a Web Page

The content on a Web page needs to be attractive. You can enhance the appearance of the content of the Web page by using formatting tags. The following formatting tags are used to format the content in an HTML document:

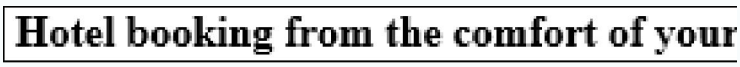
- `<B>`
- `<I>`
- `<U>`
- `<LI>`
- `<BR>`
- `<HR>`

<B>

The `<B>` tag is used to make the text boldface. Consider the following code snippet for defining the `<B>` tag:

```
<BODY>
<P><B> Hotel booking from the comfort
of your home.</B></P>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.



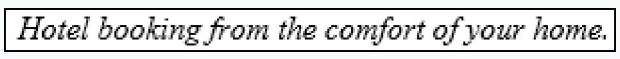
The Output Displaying the Usage of the <B> Tag

<I>

The `<I>` tag is used to italicize the text. The `<I>` tag is a container tag. Consider the following code snippet for defining the `<I>` tag:

```
<BODY>
<P><I> Hotel booking from the comfort
of your home.</I></P>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.



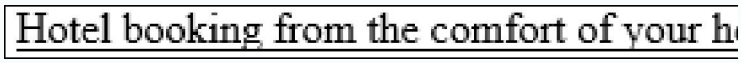
The Output Displaying the Usage of the <I> Tag

<U>

The `<U>` tag is used to underline the text. The `<U>` tag is a container tag. Consider the following code snippet for defining the `<U>` tag:

```
<BODY>
<P><U> Hotel booking from the comfort
of your home.</U></P>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.



The Output Displaying the Usage of the <U> Tag

<LI>

The <LI> tag is used to create lists. In HTML, lists are represented with the help of a special set of tags. These special tags have attributes that help in manipulating the appearance of the lists displayed on the Web pages. These tags may be nested, which means that one set of tags can be embedded within another set of tags. The following types of lists are used in HTML:

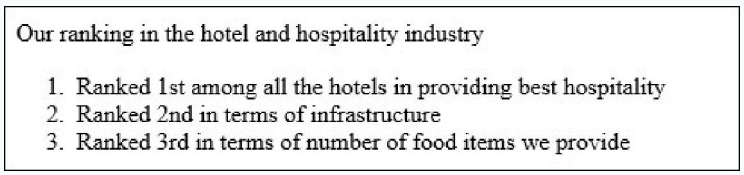
- Ordered or numbered list
- Unordered or bulleted list
- Definition list

Ordered or Numbered List

An ordered list or a numbered list represents a set of items in a sequence or an order. The tag for the ordered list is <OL>. The <OL> tag contains individual list items represented by the <LI> tag. By default, all the <LI> tags within the <OL> and </OL> tags are separated by a line break. Both, <OL> and <LI>, are container tags. Consider the following code snippet for defining the <OL> tag:

```
<BODY>
<P>Our ranking in the hotel and
hospitality industry</P>
<OL>
<LI>Ranked 1st among all the hotels
in providing best hospitality</LI>
<LI>Ranked 2nd in terms of
infrastructure</LI>
<LI>Ranked 3rd in terms of number of
food items we provide</LI>
</OL>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.



An Ordered List

The <OL> tag provides various attributes that are used to mark a list. These tags are described in the following table.

Attribute	***Value***	***Description***
reversed	*reversed*	*Specifies that the list order should be descending.*
start	*number*	*Specifies the start value of a list.*
type	*1* *A* *a* *I* *i*	*Specifies the style of the bullet to be used in a list.*

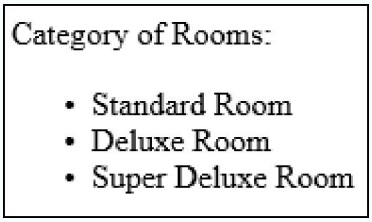
The Attributes of the <OL> Tag

Unordered or Bulleted List

An unordered or bulleted list represents a set of related items that do not need to follow a specific order. The tags used to represent the unordered list are <UL> and </UL>. The <UL> tag contains individual list items, which are represented by the <LI> tag. Consider the following code snippet for defining the <UL> tag:

```
<BODY>
<P>Category of Rooms:</P>
<UL>
<LI>Standard Room</LI>
<LI>Deluxe Room</LI>
<LI>Super Deluxe Room</LI>
</UL>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.



An Unordered List

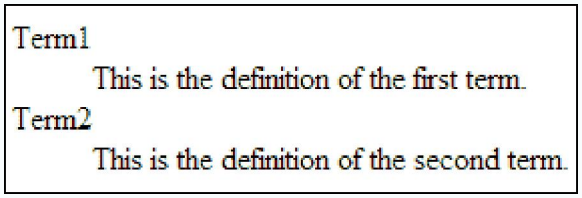
Definition List

A definition list is used when one or more terms and their definitions are to be included in an HTML document. The definition list is represented by the <DL> tag. This container tag comprises two other tags, <DT> and <DD>, where DT stands for Data Term and DD stands for Data Definition. The data term text is displayed on the left and the data definition term is displayed slightly right to it. There is a paragraph indent after the data term. The opening list tag is the <DL> tag followed by the first data term, <DT>.

Consider the following code snippet for defining the <DL> tag:

```
<BODY>
<DL>
<DT>Term1</DT> <DD>This is the
definition of the first term.</DD>
<DT>Term2</DT> <DD>This is the
definition of the second term.</DD>
</DL>
</BODY>
```

The preceding code snippet displays two terms and their definition on a Web page. The output of the preceding code snippet is displayed, as shown in the following figure.

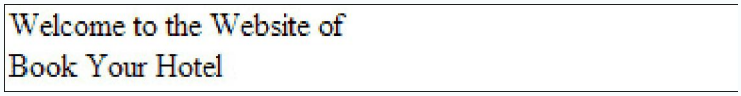


The Output Displaying the Usage of the <DL> Tag

The
 tag is used to insert a single line break. It is an empty tag, which means that it has no end tag. Consider the following code for defining the
 tag in HTML:

```
<BODY>
Welcome to the Website of<BR> Book
Your Hotel
</BODY>
```

The output of the preceding code is displayed, as shown in the following figure.



*The Output Displaying the Usage of the
 Tag*

<HR>

The <HR> tag adds a horizontal rule in a Web page. A rule is a straight line. The <HR> tag is an empty tag. Consider the following code snippet for defining the <HR> tag:

```
<BODY>
<P> Welcome to "Book Your Hotel"
website.
<HR>
<BR> This site is very user friendly.
<BR> This site gives information of
all the hotels in USA.
</P>
<P> You can book your hotel while
sitting at home. </P>
</BODY>
```

The output of the preceding code is displayed, as shown in the following figure.

The Output Displaying the Usage of the <HR> Tag

Adding Images

Images add an artistic value to a Web page. They make the Web page more interesting as ideas are well communicated with images and graphics.

Adding images to a website helps enhance its visual appeal. Also, the human brain has a tendency to recollect facts faster through visual aids, such as graphics, as compared to the use of text. The three extensively-used image formats on the Web are:

- **GIF**: The Graphic Interchange Format (GIF) format is the most commonly-used image format on the Web. It is used if the image involves drawing lines, such as images having polygonal shapes. It is considered superior to other formats for its clarity and ability to maintain the originality of an image without lowering its quality.
- **JPEG**: The Joint Photographic Experts Group (JPEG) format is used when the image is a photograph, medical image, or complex photographic illustration. JPEG images are inherently full-color images. Therefore, they appear distorted when viewed on a monitor supporting 256 colors or less.
- **PNG**: The Portable Network Graphics (PNG) format is an alternative to GIF. The PNG format defines a portable, well-compressed, and well-specified standard for bitmapped image files that retain their high resolution.

Resolution is the number of pixels that make up an image. The greater the number of pixels, the higher is the resolution.

You can use the <IMG> tag to insert images in a website. The <IMG> tag is used to place static as well as animated images on a Web page. You need to use the src attribute inside the <IMG> tag to specify the URL of the image that you want to insert in the Web page. Consider the following code snippet for defining the <IMG> tag:

```
<BODY>
<IMG src="Room.jpg">
</BODY>
```

The preceding code snippet displays the Room.jpg image by using the src attribute of the <IMG> tag. This image is stored in the same folder where the Web page in which you want to display the image is stored. However, if the image is stored at some other location, you need to specify the whole path of the image in the src attribute. The output of the preceding code snippet is displayed, as shown in the following figure.

The Output Displaying the Usage of the <IMG> Tag

While inserting images in a Web page, you can also specify their height, width, and alternative text. This can be done by using the following attributes:

- `alt`: The `alt` attribute is used to specify an alternate text for an image. This text is displayed if the image cannot be displayed in a browser. The value of the `alt` attribute is a user-defined text. Consider the following code snippet for defining the `alt` attribute:

```
<IMG src="Room.jpg"
alt="DeluxeRoom">
```

  In the preceding code snippet, the image named Room.jpg is displayed on the Web page. If the browser does not support images, the alternate text, `DeluxeRoom`, will be displayed on the Web Page.

- `height` and `width`: The `height` and `width` attributes specify the size in which the image should be displayed. These attribute values are, by default, specified in pixels. Consider the following code snippet for setting the height and width in the `<IMG>` tag:

```
<IMG src= "Room.jpg" alt="
DeluxeRoom Hotel" width="304"
height="228">
```

  In the preceding code snippet, the height and width of an image has been set to `228` and `304` pixels, respectively.

Adding Navigation Links

A website consists of interlinked Web pages. To view all the Web pages in the website, you need to navigate through the Web pages. An easy way of navigating is using hyperlinks. A hyperlink provides links to Web pages, images, multimedia files, and documents created by using applications, such as Acrobat Reader and MS Word. If the website has a poor navigational aspect, the user will move away from it. The navigation between the Web pages is important. Therefore, the Web pages should be linked logically. This can be achieved by creating a hyperlink between the Web pages by using the `<A>` tag. The `<A>` tag, which is known as an anchor tag, defines a hyperlink that links one page to another. The attributes of the `<A>` tag are:

- `href`: The `href` attribute specifies the URL of the document that opens on clicking the hyperlink. Consider the following code snippet for defining the `<A>` tag along with its `href` attribute:

```
Log on to the <A href=
"homepage.html"> BookYourHotel
</A> site to get your favorite
hotel booked.
```

  In the preceding code snippet, clicking the hyperlink, `BookYourHotel`, navigates the user to homepage.html.

- `target`: Browser windows can have names associated with them. A hyperlink in one browser window can refer to another window by using the `target` attribute of the `<A>` tag. When the link is activated, the referenced HTML document is displayed in the window specified by the `target` attribute. If the window is not open, the Web browser opens a new window. The `target` attribute is used to open the HTML document in a specified frame or window. Consider the following code snippet for defining the target attribute in the `<A>` tag:

```
<A href= "Aboutus.html" target=
"Frame1">About Us</A>
```

  In the preceding code snippet, the `Aboutus.html` document is opened in a window named Frame1 when a user clicks the hyperlink, `About Us`.

Adding Multimedia Components

Nowadays, the use of multimedia elements, such as audio and video, is common on the websites. Adding multimedia elements help you embed sound in the website and improve visual effect of the website, which, in turn, grabs the attention of the viewers. You can add multimedia elements in a Web page by using the following tags:

- `<AUDIO>`
- `<VIDEO>`

<AUDIO>

The `<AUDIO>` tag is used to include an audio file in a Web page. It provides various attributes to help you play an audio.

The attributes of the `<AUDIO>` tag are described in the following table.

Attribute	*Value*	*Description*
autoplay	*autoplay*	*Specifies that the audio starts playing as soon as it is ready to play.*
controls	*controls*	*Specifies that the controls, such as play, pause, and*

		stop, should be displayed.
src	*url*	*Specifies the URL of an audio file. It can be absolute URL or relative URL.*
loop	*loop*	*Specifies that the audio will start playing again, every time it has finished playing.*

The Attributes of the <AUDIO> Tag

Consider the following code snippet for defining the <AUDIO> tag:

```
<AUDIO controls="controls"
src="Audio1.mp3">
Your browser does not support this
audio format.
</AUDIO>
```

In the preceding code snippet, the `<AUDIO>` tag is used to display the audio file, `Audio1.mp3`, on the Web page. The content inside the `<AUDIO>` and `</AUDIO>` tags is displayed when the browser does not support the audio format.

<VIDEO>

The `<VIDEO>` tag is used to display a video file on a Web page. It provides various attributes to help you play a video.

The attributes of the `<VIDEO>` tag are described in the following table.

Attribute	*Value*	*Description*
autoplay	*autoplay*	*Specifies that the video starts playing as soon as it is ready to play.*
controls	*controls*	*Specifies that the controls, such as play, pause, and stop, should be displayed.*
src	*URL*	*Specifies the URL or location of a video file.*
loop	*loop*	*Specifies that the video will start playing again, every time it is finished playing.*
height	*pixels*	*Specifies the height of a video player in pixels.*
width	*pixels*	*Specifies the width of a video player in pixels.*
muted	*muted*	*Specifies that the audio of the video file should be muted.*
poster	*URL*	*Specifies an image to be displayed while the video is downloading.*

The Attributes of the <VIDEO> Tag

Consider the following code snippet for defining the `<VIDEO>` tag:

```
<VIDEO width="320" height="240"
controls="controls"
autoplay="autoplay" src="D:\HTML
\Video1.mp4">
Your browser does not support this
audio format.
</VIDEO>
```

In the preceding code snippet, the `<VIDEO>` tag is used to display the video file, `Video1.mp4`, on the Web page. The content inside the `<VIDEO>` and `</VIDEO>` tags is displayed when the browser does not support the video format.

Measuring Data and Displaying a Progress Bar

Consider a scenario where you need to enable a user to download files, such as images, from a website. In addition, you want that while saving the file, the user is able to view the disk space that is left. Moreover, you require that while saving the file, the user is able to know the level of the task that has been completed. In other words, the progress of the task should be displayed to the user. These tasks can be done by using the following tags:

- `<METER>`
- `<PROGRESS>`

<METER>

The `<METER>` tag specifies a scalar measurement within a known range. It is also known as gauge. It can be used to display disk usage. It provides various attributes to help you measure data. The attributes of the `<METER>` tag are described in the following table.

Attribute	*Value*	*Description*
high	*Number*	*Specifies the range considered to be the highest range.*
low	*Number*	*Specifies the range considered to be the lowest range.*
max	*Number*	*Specifies the maximum value of the range.*

min	Number	Specifies the minimum value of the range.
value	Number	Specifies the current value.
form	form_id	Specifies the ID of the form to which the <METER> tag belongs.

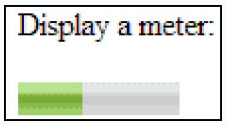
The Attributes of the <METER> Tag

Forms will be discussed in detail in the subsequent chapters of this book.

Consider the following code snippet for defining the <METER> tag:

```
<P>Display a meter:</P>
<METER value="4" min="0" max="10"></
METER>
```

The preceding code snippet displays a meter specifying that 4 out of 10 tasks have been done as the current value has been specified as 4 and the maximum value as 10. The output of the preceding code snippet is displayed, as shown in the following figure.



A Meter Being Displayed

<PROGRESS>

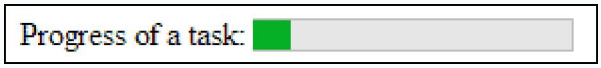
The <PROGRESS> tag is used to display the progress of a task. It provides the following attributes to display the progress bar:

- max: Specifies the amount of work a task requires.
- value: Specifies the amount of task that has been completed.

Consider the following code snippet for defining the <PROGRESS> tag:

```
Progress of a task:
<PROGRESS value="12" max="100">
</PROGRESS>
```

The preceding code snippet displays a progress bar specifying that 12% of the task has been done, as shown in the following figure.



The Progress Bar Being Displayed

Identifying Semantic Tags

Semantic tags are used to provide better readability of Web pages to Web designers. They clearly define their meaning to the browser. The semantic tags provided by HTML are:

- <HGROUP>
- <ARTICLE>
- <ASIDE>
- <HEADER>
- <FOOTER>
- <NAV>
- <FIGURE>

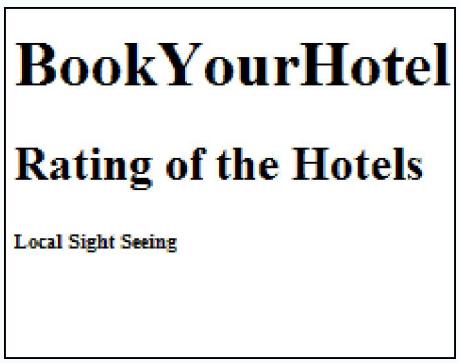
<HGROUP>

Consider a situation where you need to group related headings and sub titles together. For example, you need to show the heading and sub heading, Book Your Hotel and Rating of the Hotels, together. In such a situation, you can use the <HGROUP> tag. This tag allows you to group different levels of headings. This tag helps you associate secondary titles, such as subheadings, alternative titles, and tag lines, with the main heading.

Consider the following code snippet for defining the <HGROUP> tag:

```
<BODY>
<HGROUP>
<H1> BookYourHotel </H1>
<H2>Rating of the Hotels</H2>
<H6>Local Sight Seeing</H6>
</HGROUP>
</BODY>
```

The output of the preceding code snippet is shown in the following figure.



The Output of the Usage of the <HGROUP> Tag

<ARTICLE>

The <ARTICLE> tag defines an independent or a self-contained content. It is mostly used to specify independent entry for a blog or a magazine. Consider the following code snippet for defining the <ARTICLE> tag:

```
<BODY>
<ARTICLE>
<H1>The BookYourHotel Blog</H1>
<P>This Blog is specifically designed
to provide details about
BookYourHotel </P>
</ARTICLE>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.

The Output Displaying the Usage of the <ARTICLE> Tag

<ASIDE>

The <ASIDE> tag specifies the content other than the main tag, such as a note or a tip. However, it should be related to the main content. Consider the following code snippet for defining the <ASIDE> tag:

```
<BODY>
<H3>Book Your Hotel</H3>
<ASIDE>
<P>Get discounts on booking rooms
today</P>
</ASIDE>
</BODY>
```

The output of the preceding code snippet is displayed, as shown in the following figure.

The Output Displaying the Usage of the <ASIDE> Tag

<HEADER>

The <HEADER> tag is used to group introductory headings or navigational links. It generally includes website heading and information, such as the title and version of the website. You can use multiple <HEADER> tags in your document. Each of these tags will become a header for a particular section.

Consider the following code snippet for defining the <HEADER> tag:

```
<!DOCTYPE HTML>
<HTML>
<HEADER>
<H1>Book Your Hotel</H1>
<P>An easy way of booking hotels</P>
</HEADER>
</HTML>
```

The preceding code snippet creates a header consisting of the heading and paragraph text for a Web page.

<FOOTER>

The <FOOTER> tag is used to represent footer for a Web page or a section of a Web page. The footer generally contains the information, such as the author of the document, copyright information, privacy policy, and links to related documents. Consider the following code snippet for defining the <FOOTER> tag:

```
<!DOCTYPE HTML>
<HTML>
<FOOTER>
Author: John Bart
</FOOTER>
</HTML>
```

The preceding code snippet displays the author name of a document in footer.

<NAV>

The <NAV> tag enables you to group links created by using the <A> tag in such a way that looks more semantic and structured. All the major navigational links are generally grouped inside the <NAV> tag. The <NAV> tag is used to declare a navigational section in a Web page that links to other pages or parts of a Web page. These links enable the user to navigate the website.

Consider the following code snippet for defining the <NAV> tag:

```
<NAV>
<A href="/Hotels/">Hotels</A> |
<A href="/Rooms/">Rooms</A> |
<A href="/Booking/">Booking</A> |
<A href="/LocalPlaces/">Local
Places</A>
</NAV>
```

In the preceding code snippet, all the navigational links are placed inside the <NAV> tag. This creates a navigational section in the Web page.

The output of the preceding code snippet is displayed, as shown in the following figure.

The Output Displaying the Usage of the <NAV> Tag

<FIGURE>

The <FIGURE> tag is used to specify the self-contained content, such as images, diagrams, photos, and code, and associate caption with it. It should have some content related to the main content. However, if removed, it should not affect the flow of the document.

Consider the following code snippet for defining the <FIGURE> tag:

```
<BODY>
<FIGURE>
<IMG src="LocalPlacestoVisit.jpg"
alt="ZContinental" width="304"
height="228">
</FIGURE>
</BODY>
```

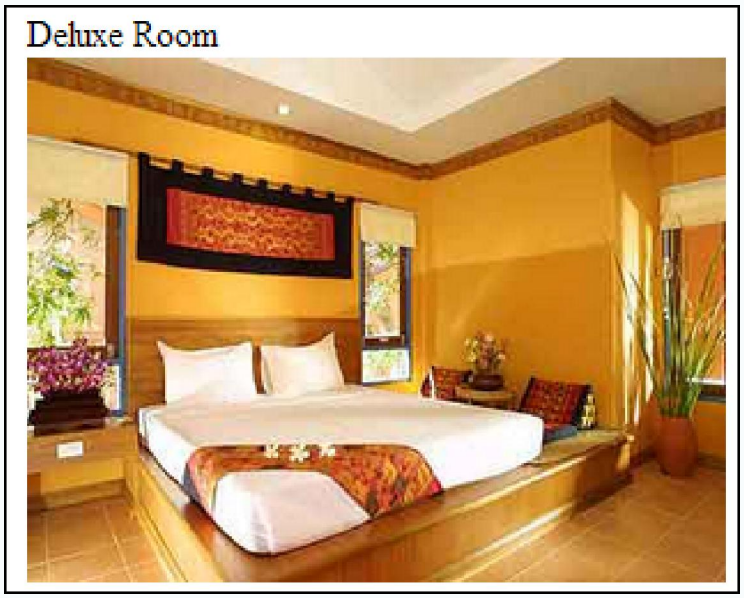
In the preceding code snippet, an image will be displayed which will give a visual glimpse of local places near the hotel. If this <FIGURE> tag is removed, the content about the hotel will not be affected because it is more important to show the pictures of the hotel rather than pictures of the local places.

You can also define a caption for the image inserted through the <FIGURE> tag by using the <FIGCAPTION> tag. The <FIGCAPTION> tag is the optional tag and can appear either before or after the content within the <FIGURE> tag. In the <FIGURE> tag, only one <FIGCAPTION> tag can be nested. Consider the following code snippet for defining the

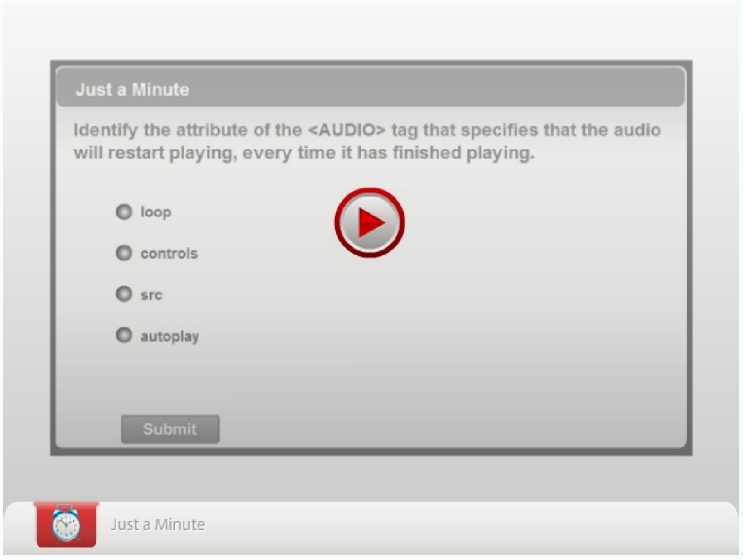
<FIGCAPTION> tag:

```
<FIGURE>
<FIGCAPTION> Deluxe Room </
FIGCAPTION>
<IMG src="Room.jpg"
alt="ZContinental" width="304"
height="228">
</FIGURE>
```

In the preceding code snippet, a figure caption, `Deluxe Room`, along with its image will be displayed on the Web page, as shown in the following figure.



The Image with the Caption Being Displayed



Activity 1.1: Creating a Web Page

Summary

In this chapter, you learned that:

- HTML is a markup language that enables you to create attractive and interactive websites.
- Web pages are either static or dynamic.
- Web pages are created by using software applications known as editors. HTML editors are broadly classified into the following categories:
  - Text editor
  - Graphic editor
- An HTML page contains the following structural tags:
  - <!DOCTYPE HTML>
  - <HTML>
  - <HEAD>
  - <BODY>
- The following tags are used in the <HEAD> section of an HTML document:
  - <TITLE>
  - <META>
  - <BASE>
  - <STYLE>
  - <LINK>
  - <SCRIPT>
  - <NOSCRIPT>
- You can use the following tags to add new paragraphs in an HTML document:
  - <P>
  - <DETAILS>
  - <SUMMARY>
- To define a layout, you can use the following tags:
  - <DIV>
  - <SPAN>
- The following formatting tags are used to format the content in an HTML document:
  - <B>
  - <I>
  - <U>
  - <LI>
  -
  - <HR>
- You can use the <IMG> tag to insert images in a website.
- You can add multimedia elements in a Web page by using the following tags:
  - <AUDIO>
  - <VIDEO>
- The <METER> tag specifies a scalar measurement within a known range.
- The <PROGRESS> tag is used to display the progress of a task.
- The semantic tags provided by HTML are:
  - <HGROUP>
  - <ARTICLE>
  - <ASIDE>
  - <HEADER>
  - <FOOTER>
  - <NAV>
  - <FIGURE>

Reference Reading

Introducing HTML

Reference Reading: Books	*Reference Reading: URLs*
Introducing HTML5 By Bruce Lawson, Remy Sharp	*http://www.w3schools.com/html/default.asp* *http://www.tizag.com/htmlT/*

Creating an HTML Web Page

Reference Reading: Books	*Reference Reading: URLs*
HTML Author: Dick Oliver Publisher: Techmedia	*http://www.w3schools.com/html/html_intro.asp* *http://www.webmonkey.com/2010/02/building_web_pages_with_html_5/*

Chapter 2

Enhancing Web Pages

A website that is professionally designed requires a lot more than just creating Web pages and linking them together. It is important that all the elements of a Web page are well formatted so that the look and feel of the Web pages should be consistent across the entire website. This can be implemented by using style sheets.

In addition, you can apply transition and transformation effects on HTML elements to add visual effects.

This chapter discusses how to implement styles by using style sheets. In addition, it discusses how to apply transitions, animations, and transformations on Web pages.

Objectives

In this chapter, you will learn to:

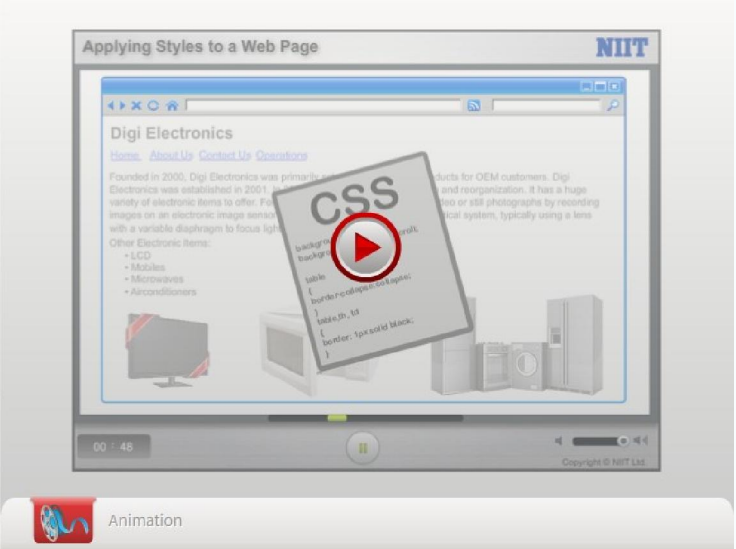
- Work with styles
- Apply transitions, animations, and transformations

Working with Styles

The look and feel of a Web page depends upon the appearance and arrangement of HTML elements on it. You can format the HTML elements in your Web page to make it look appealing. In addition, when a website is created, formatting and layout of all the Web pages should be consistent. For example, the formatting and placement of the company header and the company logo need to be same on the home page as well as on all the other Web pages of the website. To accomplish this task, Cascading Style Sheets (CSS) can be used.

CSS is a collection of styles that allows you to change the appearance of HTML elements on Web pages. It can be used to define a set of formatting options that can be used to format text and other HTML elements. It defines a set of standard rules that provides a better control over the page layout and the appearance.

CSS can be applied to specific HTML elements on a Web page, to all the elements on a Web page, or across all the Web pages of the website. Therefore, CSS can be used to enforce consistent display standards across all the Web pages of the website.



Identifying the Syntax of CSS

As a Web developer, you need to ensure that your website is visually appealing. For example, you want to color the text, background, and hyperlinks on the website. Moreover, with multiple Web pages and graphics, you want to ensure consistency in the formatting and layout of Web pages. To achieve this, you can implement CSS to stylize Web pages and text on the website.

CSS allows the creation of one or more rules for defining the style characteristics of HTML elements. Each rule consists of the following parts:

- **Selector**: A selector specifies the HTML element that you want to style. You can have one or more selectors in a rule.
- **Declaration Block**: A declaration block follows a selector in a CSS rule and is enclosed in curly braces.

The following syntax can be used to define a CSS rule:

```
selector
{
//Declaration Block
property: value;
}
```

For example, consider the following code to create the HTML document:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<H1>Welcome to BookYourHotel
website.</H1>
<H2>This site is very user
friendly.</H2>
<H1>You can book your hotel while
sitting at home.</H1>
</BODY>
</HTML>
```

Now, you want to display the text inside the `<H1>` tag in italics and red color. In addition, the text inside the`<H2>` tag should be displayed as bold. You can perform this

task by defining CSS rules using the following code snippet:

```
h1 {color:red; font-style:italic;}
h2 {font-weight:bold;}
```

In the preceding code snippet, you have created two rules to stylize the`H1` and `H2` selectors. The `color` property is given the value, `red`, and the `font-style` property is given the value, `italic`, in the first rule. Whereas, in the second rule, the `font-weight` property is given the value, `bold`. The preceding code will display the text inside the `H1` and `H2` selectors, as shown in the following figure.

The Text Inside the H1 and H2 Selectors

In addition, you can give multiple declarations inside the declaration block of a CSS rule to apply multiple styles to HTML elements. These declarations are separated by using a semi-colon. For example, the following CSS rule uses multiple declarations to stylize the body section of the preceding HTML document so that the content inside it appears in Arial font, bold and italic style, and red color:

```
body
{
font-family: arial;
font-weight: bold;
color:red;
font-style:italic;
}
```

The preceding code applies style on the body selector, as shown in the following figure.

The Styles Applied on the Body Selector

Identifying the Types of Style Sheets

The CSS styles can be applied on the websites in different ways. For example, you want to apply a unique CSS style on a specific heading of a Web page. It is also possible that you want to apply different styles on each Web page. In addition, you may want all the pages of your website to be consistent in their style. Different websites use different ways of implementing styles through CSS. Based on the manner in which CSS styles are applied, these can have the following categories:

- Inline style
- Internal style sheet
- External style sheet

Inline Style

To customize only few elements on a Web page, inline styles can be applied. For example, you want to display a particular paragraph in red color on one of the Web pages. For this, you can use an inline style. Inline styles are attached directly with the tag in the HTML document. They are specified by using the `style` attribute inside an element declaration in the HTML document. The inline styles customize only the tag on which they are applied.

Consider the following code to specify a style for only one of the `<P>` tags in the HTML document:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<P style="font-size: 24pt; color:
red">Hotel booking from the comfort
of your room.</P>
<P>Compare and book from more than
5000 hotels. </P>

</BODY>

</HTML>
```

In the preceding code, the text inside the first `<P>` element will be displayed in red color with a specific font size. Whereas, the text inside the second `<P>` element will be displayed in a normal font as no styles have been applied on it.

The following figure displays the output derived after applying the inline style on the `</P>` tag.

The Output Derived After Applying the Inline Style on `</P>` Tag

Internal Style Sheet

An internal style sheet is used when there is a need to stylize the multiple occurrences of an element on a Web page with the same style. It is also known as an embedded style sheet. For example, you want to format all the `<P>` and `<H1>` tags of a Web page. In this case, it would be advisable to define separate style rules for the `<P>` and `<H1>` tags once, and then refer to these definitions, wherever needed. The internal style sheet customizes the elements of only that Web page in which it is contained. This can be implemented by using an internal style sheet. An internal style sheet is enclosed within the `<STYLE>` tag inside the head section of the HTML document, as shown in the following code:

```
<!DOCTYPE HTML>
<HTML>
```

```
<HEAD>
<STYLE type="text/css">
p
{
color:red;
font-size:20pt;
font-style:italic;
}
</STYLE>
</HEAD>
<BODY>
<P> Hotel booking from the comfort of
your room.</P>
<P> Compare and book from more than
5000 hotels.</P>
</BODY>
</HTML>
```

In the preceding code, an internal style sheet has been used to stylize all the <P> tags defined inside the HTML document.

The text inside the <P> tags is displayed, as shown in the following figure.

The Text Inside the <P> Tags

External Style Sheet

An external style sheet is used when multiple Web pages are to be styled in the same manner to ensure the consistent look and feel across the entire website. An external style contains only the formatting rules for the desired HTML elements. Therefore, it separates the design of a Web page from its content. An external style sheet can be simultaneously linked to multiple HTML documents. Therefore, a consistent style can be applied on multiple pages of the website by defining an external style sheet.

An external style sheet is a text document that consists of CSS formatting rules. This document can be written in a simple text editor, such as Notepad, and then saved with the .css file extension. To associate a Web page with an external style sheet, you need to use the <LINK> tag inside the head section of the HTML document. The following code displays the <LINK> tag along with its some commonly-used attributes:

```
<LINK type="text/css"
rel="stylesheet"
href="externalstylesheet.css" />
```

In the preceding syntax:

- `type`: Specifies the type of content in the linked document.
- `href`: Specifies the location of the linked external style sheet.
- `rel`: Specifies the relationship between the CSS document and the HTML document. The `rel` attribute is specified with value, `stylesheet`, which specifies that the current HTML document is importing an external style sheet.

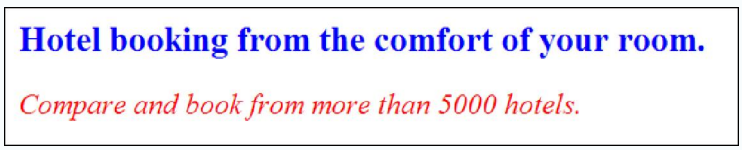
Consider the following code to associate an external style sheet with the HTML document:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<TITLE> An External Style Sheet </
TITLE>
<LINK type="text/css"
rel="stylesheet"
href="externalstyle.css" />
</HEAD>
<BODY>
<H1> Hotel booking from the comfort
of your room. </H1>
<P> Compare and book from more than
5000 hotels. </P>
</BODY>
</HTML>
```

The <LINK> tag in the preceding code associates the external style sheet named `externalstyle.css` with the HTML document. The content of the file, `externalstyle.css`, is shown in the following code snippet:

```
p
{
color:red;
font-size:20pt;
font-style:italic;
}
h1
{
color:blue;
font-size:25pt;
font-weight:bold;
}
```

The Web page associated with the external style sheet is displayed, as shown in the following figure.



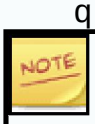
The Web Page Associated with the External Style Sheet

Applying Multiple Style Sheets

Consider an example where you have defined different formatting rules for the <P> tag in the form of an inline style, as well as in the form of an internal style in the same HTML document. Therefore, properties have been set for the same selector in multiple style sheets. In such a case, the style that is most specific to the element and has the highest priority will be used to stylize the

element. The following list displays the priority of style sheets in descending order:

- Inline style
- Internal style sheet
- External style sheet
- Browser default



The browser default is the default style applied on HTML elements, if no style type has been defined.

Consider the following code snippet for the `externalstylesheet.css` file:

```
p{
color:blue;
font-size:12pt;
font-weight:bold;
}
```

Consider the following code to associate multiple style sheets with the HTML page:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<LINK type="text/css"
rel="stylesheet"
href="externalstylesheet.css" />
<STYLE>
h1{
color:red;
font-size:12pt;
font-style:italic;
}
</STYLE>
</HEAD>
<BODY>
<P>Welcome to BookYourHotel
website.</P>
<H1 style="font-size: 24pt; color:
green"> Hotel booking from the
comfort of your room.</H1>
<H1> Compare and book from more than
5000 hotels. </H1></BODY>
</HTML>
```

In the preceding code, the first heading will appear in green with font size, 24, since the inline style defined for the first `<H1>` tag will override the internal style defined in the `<STYLE>` tag. However, the second heading will appear in italics, red color with font-size, 12, according to the internal style defined for the `<H1>` tag. The paragraph will be styled according to the external style sheet as no internal or inline style has been applied on it.

The Web page with multiple styles is displayed, as shown in the following figure.

Identifying CSS Selectors

Consider an example where you want to stylize a specific `<P>` tag differently from the rest of the paragraphs in the same Web page. Therefore, this `<P>` tag needs to be uniquely identified for the application of a distinct style. On the other hand, you may want to specify a common style for different elements, such as headings, lists, and paragraphs, on a Web page. In both these cases, you need to create your own CSS selectors. These user-defined CSS selectors are classified as ID and class selectors.

Implementing the ID Selector

CSS styles can be applied to an element with a specific ID by using an ID selector. An ID selector is used to identify an element that you need to style differently from the rest of the page. An ID selector is defined by using the hash symbol (#).

Consider a scenario where you need to specify a unique style for the paragraph that displays the welcome message on your website. This can be accomplished by adding an ID attribute to the paragraph that displays this message and using that ID to specify the style that needs to be applied to this paragraph.

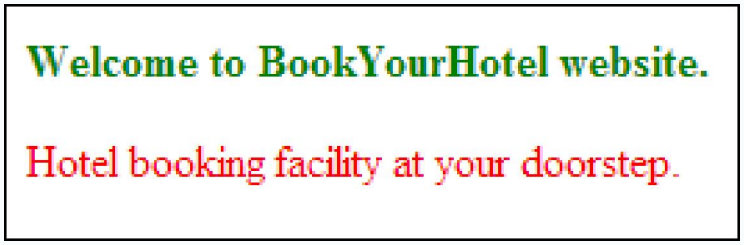
Consider the following code to use an ID selector:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
p
{
color:red;
}
#pname
{
color:green;
font-size=20;
font-weight:bold;
}
</STYLE>
</HEAD>
<BODY>
<P ID="pname">Welcome to
BookYourHotel Website.</P>
<P> Hotel booking facility at your
doorstep</P>
</BODY>
<HTML>
```

In the preceding code, the first `<P>` tag is given the ID, `pname`, which is being used as the ID selector in the `<STYLE>` tag to define a style rule for this `<P>` tag. Therefore, the first `<P>` tag will be stylized as per the ID selector, `#pname`. However, the second `<P>` tag will appear in red color as defined by the internal style sheet.

The output derived after using an ID selector is

displayed, as shown in the following figure.



The Output Derived After Using an ID Selector

Implementing the Class Selector

A CSS style can be applied to a group of elements by using the class selector. The class selector is used when there is a need to apply the same style on different types of elements in the HTML document. Multiple elements can belong to the same class. Hence, you can set a particular style for many HTML elements with the same class. The class selector is defined by using a dot (.).

Consider an example of a Web page where you need to apply the same formatting rules on paragraphs and headings of the Web page. You can accomplish this by using the internal style sheets, as shown in the following code:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
p{
font-size:20px;
color: red;
font-weight:bold;
}
h1{
font-size:20px;
color: red;
font-weight:bold;
}
h2{
font-size:20px;
color: red;
font-weight:bold;
}
</STYLE>
</HEAD>
<BODY>
<H1>Welcome to BookYourHotel
website.</H1>
<H2>This site is very user
friendly.</H2>
<P>This site gives information of all
the hotels in USA.</P>
<P>You can book your hotel while
sitting in your room.</P>
</BODY>
</HTML>
```

Although the preceding code gives the desired output, it has certain drawbacks associated with it. The code is redundant as the same style has been defined more than once for different selectors. Therefore, it makes the code bulky. In such a case, instead of defining the same style repeatedly for different elements, you should use a class selector. All the elements with the same class can be assigned the same style. Consider the following code snippet to create a class selector for styling various HTML elements of the Web page:

```
.first
{
font-size:20px;
color: red;
font-weight:bold;
}
```

In the preceding code snippet, a class selector has been created. This selector can be associated with the desired HTML elements by using the `class` attribute, as shown in the following code snippet:

```
<BODY>
<H1 class="first"> Welcome to
BookYourHotel website.</H1>
<H2 class="first"> This site is very
user friendly. </H2>
<P class="first"> This site gives
information of all the hotels in
USA.</P>
<P class="first"> You can book your
hotel while sitting at your room.</P>
</BODY>
```

In addition, you can also set a class selector in such a way that only specific HTML elements should get affected by a class. You can do this by placing the type selector before the dot of the class selector.

Consider an example where you want HTML elements in your Web page to be colored blue. This can be implemented by creating a class selector. However, you also want some specific <P> tags to be colored red. This can be done by creating a class selector specifically for these paragraphs. Its name is preceded by the type selector, `p`. Consider the following code snippet to specify that only specific <P> tags should be affected by the style of the class selector:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
.color1 {
color:blue;
}
p.color2 {
color:red;
}
</STYLE>
</HEAD>
<BODY>
<H4 class="color1"> Welcome to
BookYourHotel website.</H4>
<P class="color1">Provides online
booking of domestic and international
hotels.</P>
```

```
<P class="color2"> Avail great
discounts and offers. </P>
</BODY>
</HTML>
```

The output derived after using a class selector is displayed, as shown in the following figure.

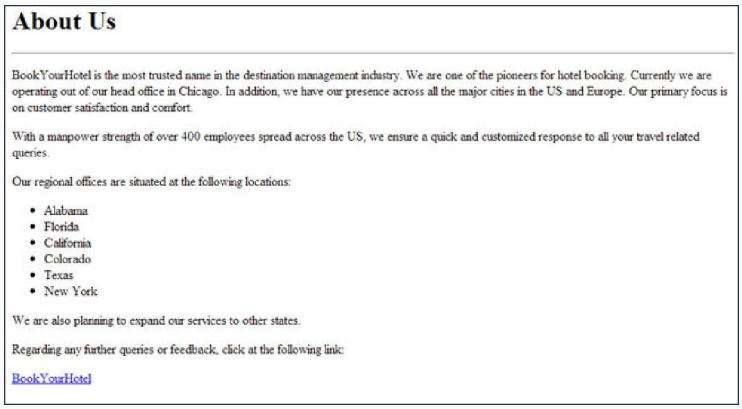
The Output Derived After Using a Class Selector

Styling HTML Elements

Consider the scenario of the BookYourHotel website. The following code is used to create the About Us page of the website:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
</HEAD>
<BODY>
<H1>About Us</H1><HR>
<P>BookYourHotel is the most trusted
name in the destination management
industry. We are one of the pioneers
for hotel booking. Currently we are
operating out of our head office in
Chicago. In addition, we have our
presence across all the major cities
in the US and Europe. Our primary
focus is on customer satisfaction and
comfort.</P>
<P>With a manpower strength of over
400 employees spread across the US,
we ensure a quick and customized
response to all your travel related
queries.</P>
<P>Our regional offices are situated
at the following locations:
<UL>
<LI>Alabama</LI>
<LI>Florida</LI>
<LI>California</LI>
<LI>Colorado</LI>
<LI>Texas</LI>
<LI>New York</LI>
</UL>
<P>We are also planning to expand our
services to other states.</P>
<P>Regarding any further queries or
feedback, click at the following
link:</P> <A
href="contactus.html">BookYourHotel</
A>
</BODY>
</HTML>
```

The About Us page of the BookYourHotel website is displayed, as shown in the following figure.



The About Us Page of the BookYourHotel Website

To improve the look and feel of this page, you need to incorporate the following style guidelines:

- The text in the paragraphs should have the font family as Helvetica and the font size as 20.
- The Heading level one should appear in small caps.
- The background should appear in grey color.
- The space between letters in a paragraph should be 2 points. Also, the line height should be 12 points.
- The symbol of the list bullets should appear as a square.

The preceding style guidelines can be applied by setting the corresponding CSS properties. A CSS property represents a characteristic of the HTML element that can be customized. You need to use these CSS properties to define styles for HTML elements. The CSS properties can be divided into the following categories:

- Font
- Text
- Link
- List
- Background

Font

Font properties are used to customize the manner in which some text is displayed on a Web page and also to make it more attractive. The following table lists the available font properties.

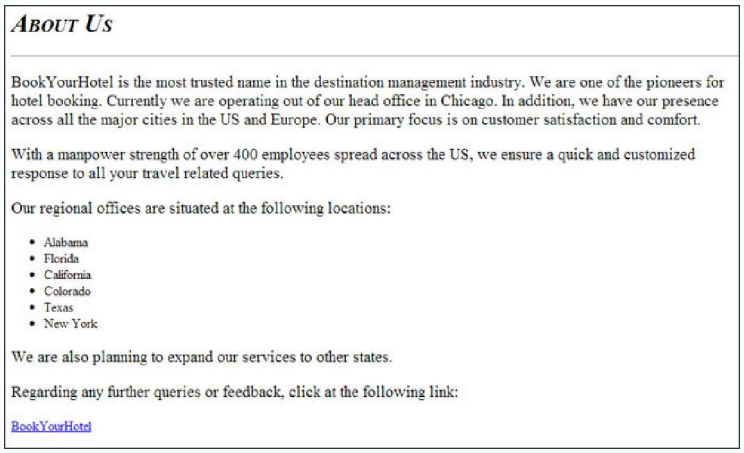
Property	Definition	Valid Values	Sample Usage
font-family	Is used to set the font type for a text.	Different font types, such as times new roman, courier, verdana, helvetica, arial, and san-serif	{font-family: arial, times roman; }
font-size	Is used to set the size of a text.	font size may be specified in px, percent, or in absolute units (large or small)	{font-size:12px; } {font-size:100%; } {font-size:x-small; } {font-size:x-large; }
font-style	Is used to set the style for a text.	normal, italic, or oblique	{font-style:italic; }
font-variant	Is used to specify whether a text should be displayed in small caps font or in a normal font.	normal or small-caps	{font-variant:small-caps; }
font-weight	Specifies how the characters in a text should be displayed.	normal or bold or decimal values from 100 to 900	{font-weight:bold; } {font-weight: " 600 "; }
font	Is used to set all the properties in one declaration.	font-style || font-variant || font-weight || font-size || font-family	font:italic bold 15px arial;

The Font Properties

In the BookYourHotel website, the existing About Us page is formatted by applying certain style for the font, as shown in the following code snippet:

```
p{
font-family: "Helvetica"
Font-size:20px;
}
h1{
font-variant:small-caps;
font-style:italic;
}
```

In the preceding code snippet, the font family and the font size of the text in the paragraph are changed according to the specifications. In addition, the heading is displayed as italicized and in small caps on the Web page. The output derived after applying the font styles is displayed, as shown in the following figure.



The Output Derived After Applying the Font Styles

Text

The CSS text properties can be used to change color, indentation, and alignment of text elements on a Web page. The following table lists the text properties that can be used to control the alignment of the text.

Property	Definition	Valid Values	Sample Usage
text-align	Is used to set the horizontal alignment of the HTML element.	left | right | center | justify	{text-align:center; }
text-indent	Is used to set the indentation of the first line in a block of text.	length in pixels | percentage	{text-indent:25px; } {text-indent:5%; }
line-height	Is used to specify the height of a line.	normal | number | length in pixels | percentage	{line-height:10pt; } {line-height:100%; } {line-height:normal; }
direction	Is used to specify the text direction, i.e. left to right or right to left.	ltr| rtl	{direction:rtl;}
vertical-align	Is used to set the vertical alignment of the HTML element.	(sub | super |baseline)|| length in pixels||percentage	{vertical-align:sub; } {vertical-align:50%; } {vertical-align:-30px; } {vertical-align:20px; }

The Text Properties Used to Control the Alignment of the Text

The following table lists the text properties that can be used to control the spacing and formatting of the text.

Property	Definition	Valid Values	Sample Usage
color	Is used to set the color of text.	A HEX value, an rgb value, or a color name.	{color:red;} {color:#00ff00;} color:rgb(0,0,255);
text-decoration	Is used to specify the decorations that can be added to a text.	none | underline | overline | line-through	{text-decoration:underline; }
text-transform	Is used to control the capitalization of text.	capitalize | uppercase | lowercase | none	{text-transform:uppercase; }
letter-spacing	Is used to set the spacing between the characters in a text.	normal | length where, length specifies an extra space between characters. Negative values are also allowed.	{letter-spacing:4pt; } {letter-spacing:-2pt; }
word-spacing	Is used to set the spacing between the words in a text.	normal | length	{ word-spacing:30px; }

The Text Properties

In the BookYourHotel scenario, you can use the following code snippet to achieve the desired effect in the textual appearance of paragraphs:

```
p{
letter-spacing:2pt;
line-height:12pt;
}
h1{
color: red;
text-align: center;
}
```

The preceding code snippet stylizes the text of the paragraphs. In addition, it aligns the heading in red color and in center. The output derived after applying the text styles is displayed, as shown in the following figure.

The Output Derived After Applying the Text Styles

Link

The CSS Link properties are used to customize the appearance of links in the HTML document. A link can attain one of the following states in the HTML document:

- `link`: An unvisited link
- `visited`: A visited link
- `hover`: A link as it appears when the mouse is placed or moved over it
- `active`: A link as it appears when it is clicked

In the BookYourHotel scenario, consider the following CSS code snippet to make a link appear in red color on hover and to have a blue background and underline, when clicked:

```
<STYLE>
a:link {
color:#FF0000;
}
a:visited {
color:#00FF00;
}
a: hover {
color: #0000FF;
}
a: active {
color: #FF00FF;
text-decoration:underline;
}
</STYLE>
```

To apply formatting to a link in the hover state, the link and its visited state must be defined first; and to ensure that formatting can be applied on a link in the active state, the hover state must be defined first in the CSS definition.

List

The CSS list properties are used to customize the look of the ordered and unordered HTML lists. Usually, the items in an unordered list are marked by using square or circular bullets. Using the CSS list properties, you can customize the marker for an unordered or ordered list. You can specify an image as a marker for an unordered list and you can specify different characters as marker for an ordered list. The following table lists the various CSS list properties.

Property	Definition	Valid Values	Sample Usage
`list-style-position`	*Is used to specify the position of list-item markers according to the content flow.*	`inside | outside`	`{ list-style-position:inside; }`
`list-style-type`	*Is used to specify the type of the list-item markers.*	`disk | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | lower-latin | upper-latin |none`	`{ list-style-type:upper-alpha; }`
`list-style-image`	*Is used to specify an image on list-item markers.*	`url | none`	`{ list-style-image:url('plus.gif'); }`
`list-style`	*Sets all the properties in one declaration.*	`type || position || image`	`{ list-style: square outside url(plus.gif); }`

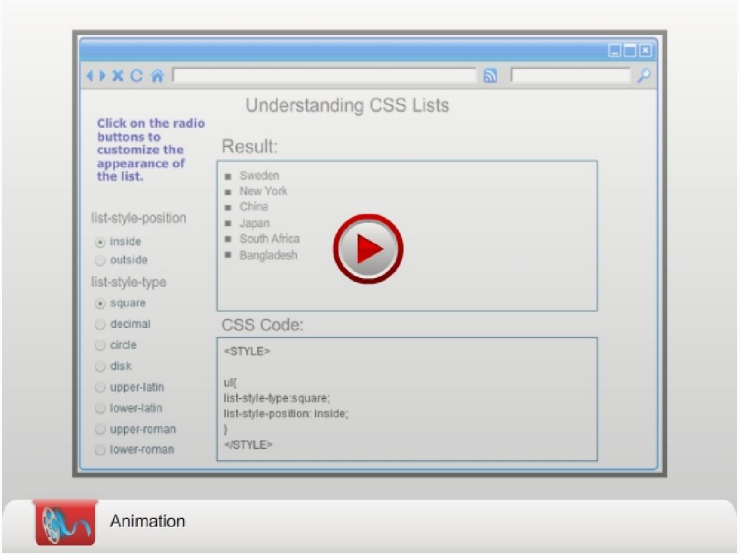
The List Properties

In the BookYourHotel scenario, consider the following code snippet to achieve the desired effect in the unordered list appearance:

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

The preceding code snippet will stylize the list-item markers in square and with the position, `inside`, as shown in the following figure.

The List-item Markers in Square and Inside Position



Background

The CSS background properties are used to specify the background color, image, or position of the HTML

element. The following table lists the background properties that can be used to set the background effects on the HTML element.

Property	Definition	Valid Values	Sample Usage
background-color	Is used to set the background color of an HTML element.	color|transparent	{background-color:yellow;}
background-image	Is used to set the background image of an HTML element.	none| url	{background-image:url('img.gif');}
background	Sets all the properties in one declaration.	background-color|background-image	{ background:green url('img.gif')};

The Background Properties

In the BookYourHotel scenario, consider the following code snippet to set the background color of the About Us page:

```
<STYLE>
body{
background-color:lavender;
}
</STYLE>
```

The preceding code snippet will stylize the About Us page in lavender color, as shown in the following figure.

The About Us Page in Lavender Color

Grouping and Nesting Styles

Consider a scenario where you want all your heading levels to have the same color. Instead of making a CSS rule for each heading separately or applying a class selector on each heading separately, you can rather group them. Similarly, you may also want to apply the same foreground color on the paragraphs and the second level headings in a Web page. For this, instead of writing the same styling code once for the paragraph tag and again for the heading tag, CSS provides you the functionality to group HTML elements. With the help of grouping, you can apply the same style on more than one HTML element without repeating them in the style sheet.

However, providing too many IDs or class selectors might become confusing. This can also make the HTML code complex. Therefore, to check such issues, you can use nesting selectors, which assign a style to an element within an element. Grouping or nesting styles provide an optimized way to write code for styling HTML elements.

Consider the following code to create the home page of the BookYourHotel website:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
h1{
text-align:center;
}
</STYLE>
</HEAD>
<BODY>
<H1>BookYourHotel</H1>
<UL>
<LI><A href="aboutus.html">About Us</
A></LI>
<LI><A href="rooms.html">Rooms</ A></
LI>
<LI><A
href="facilities.hmtl">Facilities</
A></LI>
<LI><A href="contactus.html">Contact
Us</A></LI>
</UL>
<HR>
<H3> Welcome to the Home page</H3>
<P> BookYourHotel welcomes you with
warmth and a feeling that each guest
is truly special. It offers a cozy
and intimate experience amidst the
glitz and glamour of South America.
Our caring and courteous staffs are
ever eager to ensure that all
individual needs are cared for with
professional expertise and a personal
touch.</P>
<P>The rooms have been designed with
different floor plans to create
individual spaces. We have Deluxe
Rooms, Double and Single Rooms each
with ensuite Bathrooms fitted with
all modern conveniences. All room has
Air-Conditioning and is equipped with
well stocked Mini-bars, Television
with Cable T.V. Conference facilities
available for approximately 40
persons. Our multi-cuisine restaurant
offers, a plethora of tasty local and
international dishes to satisfy all
palates. This with the elegant
ambience makes for a truly unique
experience.</P>
</BODY>
</HTML>
```

The home page of the BookYourHotel website is displayed, as shown in the following figure.

The Home Page of the BookYourHotel Website

To customize the look and feel of this page, you need to incorporate the following style guidelines:

- The links inside the list should be displayed in red color and horizontal way.
- The heading three should be positioned at the top left corner of the Web page.
- The text inside the heading one and paragraphs should be colored red.

Grouping Styles

Grouping styles are used to apply the same styles on more than one selector by combining them into a single group. The selectors in this group are separated with commas. The following syntax is used to group the elements for applying a common style:

```
selector1, selector2
{
property:value;
}
```

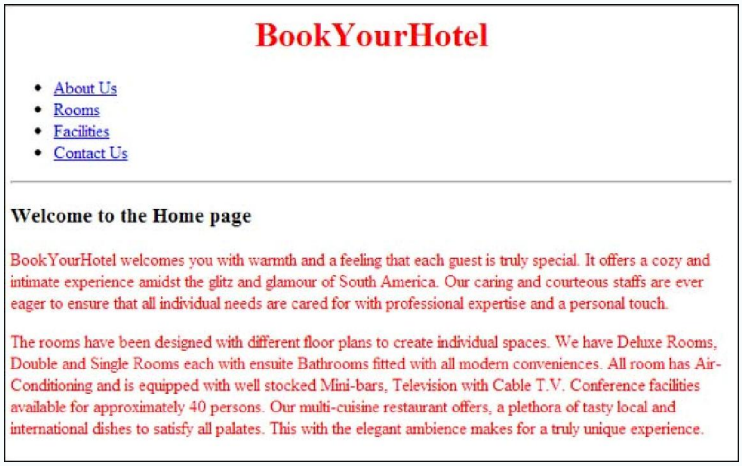
In the preceding syntax:

- `selector1`: Specifies the name, id, or class of the first element to be stylized.
- `selector2`: Specifies the name, id, or class of the second element to be stylized.
- `property`: Specifies the attribute name of an element. An element can be stylized by customizing its attributes.
- `value`: Specifies the value of the property.

Consider the following code snippet to stylize the text of both, the `H1` element and the `P` element, in same color, with grouping styles:

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

In the preceding code snippet, all text displayed by using the `H1` and `P` elements get stylized with red color, as shown in the following figure.



The H1 and P Elements Stylized with Red Color

Nesting Styles

Consider a Web page where a list or a paragraph is nested inside the div element. You may want to apply styles to the nested elements to enhance its look and feel. Styles can be applied to the nested elements by using the class or ID selectors. However, using such selectors may sometimes increase the size of the code or add to its complexity. Therefore, to achieve code optimization, CSS introduces nesting of styles. With nesting styles, you can apply style for an element within an element. It is an economic way of styling elements within an element, which discards the usage of class or ID selectors in the code.

You can apply various nesting styles on HTML elements. The following table lists the nesting styles.

Selector	***Example***	***Description***
element1>element 2	`div>p`	*Selects all the <P> elements, where <DIV> is their immediate - parent element.*
element1 element2	`div p`	*Selects all the <P> elements inside the <DIV> element.*
element1 + element2	`div+p`	*Selects the <P> element that is placed immediately after the <DIV> element.*
element1 ~ element2	`div~p`	*Selects all the <P> elements that follow the <DIV> element.*

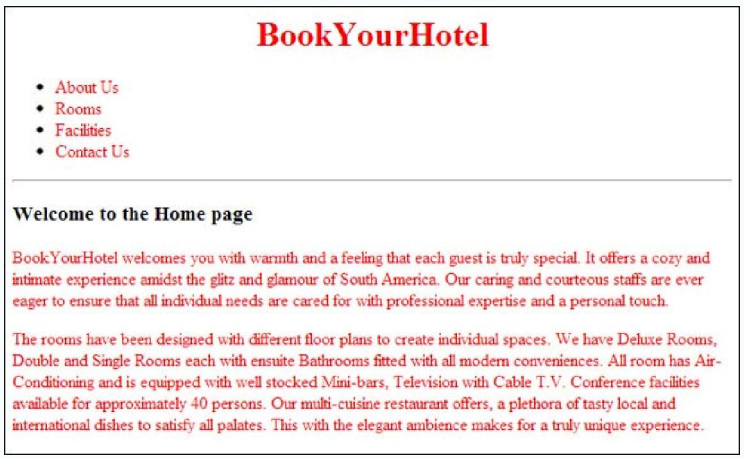
The Nesting Selectors

In the home page of the BookYourHotel website, the `<A>` tag is nested inside the `<LI>` tag. Therefore, to stylize the `<A>` tag, you can apply nesting styles. Consider the following code snippet to apply nested

styles on <A> elements:

```
li a{
color:red;
text-decoration:none;
}
```

In the preceding code snippet, links inside the list element get stylized in color red, with no text decoration, as shown in the following figure.

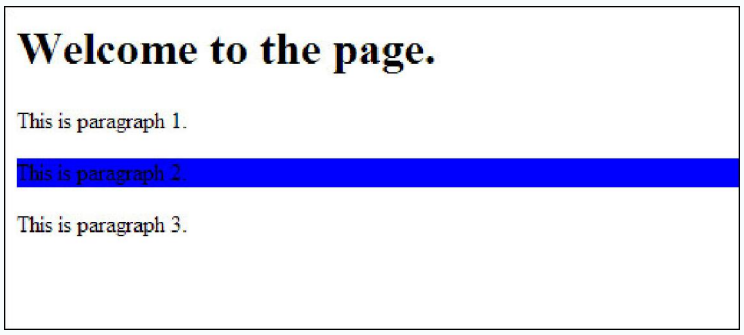


The Links Inside the List Elements Stylized in Red Color

Consider the following code snippet to stylize the paragraph tag that is placed immediately after the <DIV> tag:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
div+p
{
background-color:blue;
}
</STYLE>
</HEAD>
<BODY>
<H1>Welcome to the page.</H1>
<DIV>
<P>This is paragraph 1.</P>
</DIV>
<P>This is paragraph 2.</P>
<P>This is paragraph 3.</P>
</BODY>
</HTML>
```

In the preceding code, only the <P> tag that is placed immediately after the <DIV> tag gets stylized in blue color. The output of the preceding code snippet is shown in the following figure.



The Output of the Preceding Code Snippet

Controlling the Visibility of Elements

Consider an example where you have created a menu bar on a Web page. You want to control the visibility of the submenus in such a way that they should be visible only when the user moves the mouse pointer on the corresponding top-level menu item. You can perform this by defining visibility styles on HTML elements. CSS visibility styles can have the following categories:

- Display
- Visibility

Display

The `display` property is used to set the appearance of an element on a Web page. The following syntax can be used to apply the display property:

```
display: none| block| inline
```

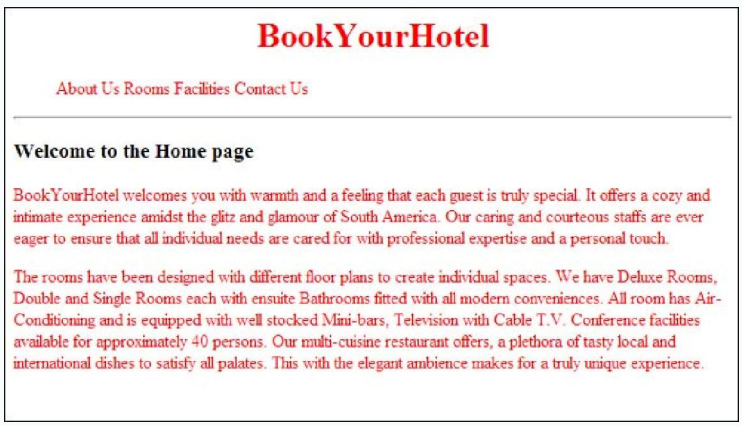
In the preceding syntax:

- `none`: Hides an element from a Web page without consuming any space.
- `block`: Shows the elements by consuming the full width available. It has a line break before and after it. Here, elements flow down the page in the normal flow.
- `inline`: Shows elements by consuming as much width as necessary. It does not have line breaks. Here, elements are laid out within the line. It is the default value that is set to the `display` property.

In the BookYourHotel scenario, consider the following snippet code to horizontally display the links inside the list on the home page:

```
li{
display: inline;
list-style-type:none;
}
```

In the preceding code snippet, links are horizontally displayed, as shown in the following figure.



The Links Displayed Horizontally

Visibility

The`visibility` property is used to specify whether an element should be visible or not. The following

syntax can be used to apply the `visibility` property:

```
visibility: hidden| visible
```

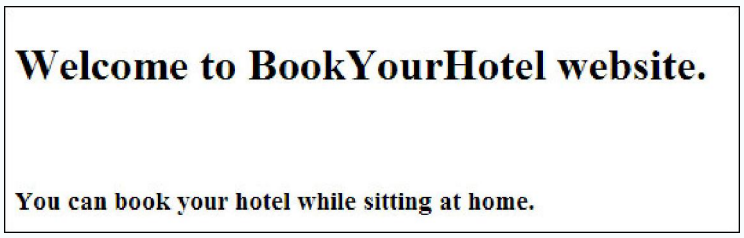
In the preceding syntax:

- `hidden`: Hides an element from a Web page. However, it will consume the space occupied by that element.
- `visible`: Shows the element on a Web page. It is the default value assigned to the visibility property.

Consider the following code snippet to use the`visibility` property:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
h2{
visibility:hidden;
}
</STYLE>
</HEAD>
<BODY>
<H1>Welcome to BookYourHotel website.
</H1>
<H2>This site is very user
friendly.</H2>
<H3>You can book your hotel while
sitting at home.</H3>
</BODY>
</HTML>
```

The preceding code will hide the text inside the `<H3>` tag. However, it will occupy the space required to display the heading. The output of the visibility property is displayed, as shown in the following figure.



The Output of the Visibility Property

Positioning HTML Elements

In HTML, elements flow one after another in the same sequence as they appear in the code. This is known as static positioning. However, you may need to change the default positioning of the elements in certain cases. For example, you need to place an element behind the other to show some overlapping effect. HTML, however, does not allow users to control the positioning of the elements on a page. This functionality of controlling the placement of elements on a Web page can be implemented by using CSS. The styles for the placement of elements can have the following categories:

- `Position`
- `Float`

Position

The `position` property is used to position an element on a Web page. The following table displays the positioning methods that can be used to position the HTML element.

Positioning Method	***Description***
`static`	*In static positioning, elements are positioned according to the normal flow of the page.*
`fixed`	*In fixed positioning, elements are positioned relative to the browser window. They will not change their position even if the browser window is scrolled.*
`relative`	*In relative positioning, an HTML element is positioned relative to its normal position.*
`absolute`	*In absolute positioning, an element can be placed at a fixed position on the Web page. Its position will not be affected by the flow of other elements.*

The Positioning Methods

With the help of the `position` property, you can specify the type of the positioning method used for the HTML element. Further, you can also define the exact location for HTML elements by using the positioning properties. The following table lists the CSS positioning properties.

Property	*Definition*	*Valid Values*	*Sample Usage*
`bottom`	*Is used to set the bottom margin for the positioned element.*	`auto|length|Percentage`	`{bottom:5px;}`
`top`	*Is used to set the top margin for the positioned element.*	`auto|length|percentage`	`{top:5px;}`
`left`	*Is used to set the left margin for the positioned element.*	`auto|length|Percentage`	`{left:5px;}`
`right`	*Is used to set the right margin for the positioned element.*	`auto|length|percentage`	`{right:5px;}`
`position`	*Is used to specify the positioning method to be used for an element.*	`static|absolute|fixed|relative`	`{position:static;}`
`z-index`	*Is used to specify whether an element will appear in front of another element or besides another element.*	`auto|number`	`{z-index:-1;}`

The CSS Positioning Properties

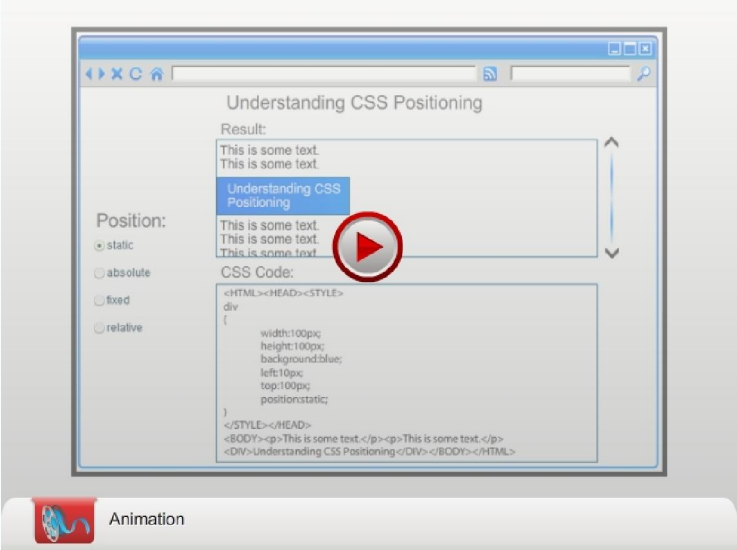
The `top`, `left`, `right`, `bottom`, and `z-index` properties cannot be used with the static positioning method.

In the BookYourHotel scenario, consider the following code snippet to position the `<H3>` tag at the top of the page:

```
h3{
position: absolute;
top:0px;
}
```

In the preceding code snippet, the text inside the <H3> tag is positioned at the top of the Web page, as shown in the following figure.

The Text Inside the <H3> Tag Positioned at the Top of the Web Page



Float

The `float` property is used to place HTML elements to the left or right margin, in relation to the other HTML elements. It allows you to wrap the HTML elements around the floated element. The concept of the float property is similar to magazines, where photos are aligned to one side, while the paragraph or text flows to the other side. The following syntax is used to specify the float property:

```
float: left|right|none
```

In the preceding syntax:

- `left`: Sets the element to the left.
- `right`: Sets the element to the right.
- `none`: Specifies that the element does not float. It is the default value given to the float property.

Consider an example of the BookYourHotel website. You have added an image to the Home page and it is currently formatted, as shown in the following figure.

The Home Page

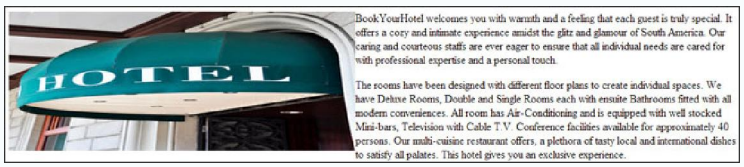
Now, you want to place the paragraphs to the right of the image. For this, consider the following code to float an image with text wrapped around it:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
img {
float: left;
}
</STYLE>
</HEAD>
<BODY>
<IMG src= "Hotel-entrance.jpg"
width=500 height=200>
<P> BookYourHotel welcomes you with
warmth and a feeling that each guest
is truly special. It offers a cozy
and intimate experience amidst the
glitz and glamour of South America.
Our caring and courteous staffs are
ever eager to ensure that all
individual needs are cared for with
professional expertise and a personal
touch.</P>
<P>The rooms have been designed with
different floor plans to create
individual spaces. We have Deluxe
Rooms, Double and Single Rooms each
with ensuite Bathrooms fitted with
all modern conveniences. All room has
Air-Conditioning and is equipped with
well stocked Mini-bars, Television
with Cable T.V. Conference facilities
available for approximately 40
persons. Our multi-cuisine restaurant
offers, a plethora of tasty local and
international dishes to satisfy all
palates. This hotel gives you an
exclusive experience.</P>
</BODY>
```

```
</HTML>
```

In the preceding code, the image is made to float to the left margin. Therefore, the text that appears after the image will flow to the right side of the image.

The output displaying the use of the `float` property is shown in the following figure.



The Output Displaying the Use of the Float Property

Clear

Whenever you apply the `float` property on any HTML element, all the elements after the floating element will be placed around it. This can be avoided by using the `clear` property. The `clear` property is used to turn off the float effect on HTML elements. The following syntax can be used to apply the `clear` property:

```
clear: both|left|right;
```

In the preceding syntax:

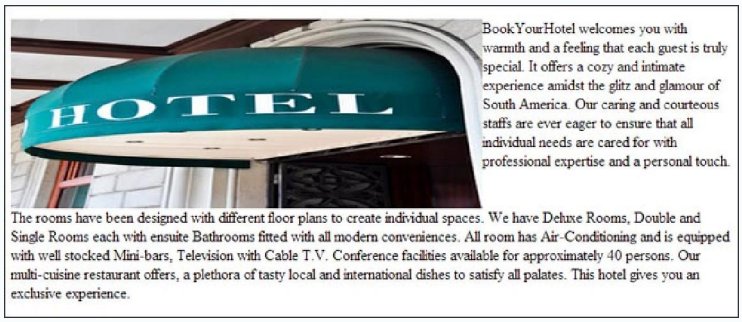
- `both`: Clears float from either direction.
- `left`: Clears float from the left direction.
- `right`: Clears float from the right direction.

You may want one paragraph to be displayed on the right side and another to be displayed below the image. This can be achieved by using the `clear` property.

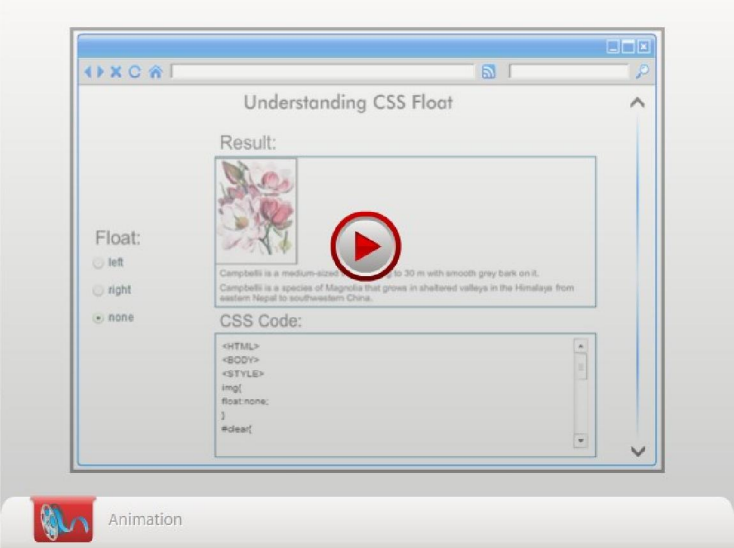
Consider the following code snippet for applying the `clear` property:

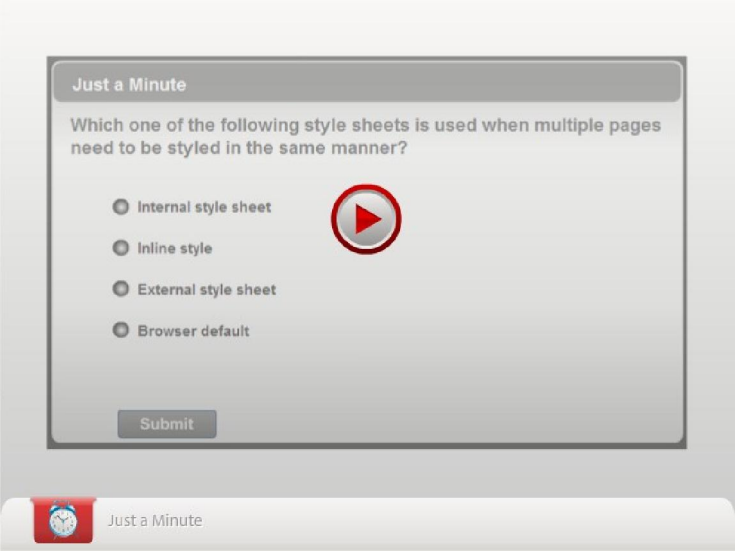
```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
img {
float: left;
}
.auto-style{
clear:both;
}
</STYLE>
</HEAD>
<BODY>
<IMG src= "Hotel-entrance.jpg"
width=500 height=200>
<P> BookYourHotel welcomes you with
warmth and a feeling that each guest
is truly special. It offers a cozy
and intimate experience amidst the
glitz and glamour of South America.
Our caring and courteous staffs are
ever eager to ensure that all
individual needs are cared for with
professional expertise and a personal
touch.</P>
<P class="auto-style">The rooms have
been designed with different floor
plans to create individual spaces. We
have Deluxe Rooms, Double and Single
Rooms each with ensuite Bathrooms
fitted with all modern conveniences.
All room has Air-Conditioning and is
equipped with well stocked Mini-bars,
Television with Cable T.V. Conference
facilities available for
approximately 40 persons. Our multi-
cuisine restaurant offers, a plethora
of tasty local and international
dishes to satisfy all palates. This
hotel gives you an exclusive
experience.</P>
</BODY>
</HTML>
```

The preceding code displays a paragraph on the right side and another paragraph below the image on the Web page. The output displaying the use of the `clear` property is shown in the following figure.



The Output Displaying the use of the Clear Property





Activity 2.1: Implementing Styles on HTML Elements

Applying Transitions, Animations, and Transformations

MySchool Pvt. Ltd. is an organization that offers Web-based learning solutions to the KinderGarten kids. The organization wants to provide learning to the students in an easy and interesting way. They make the students learn concepts, such as identifying colors and shapes, by using various animations and transitions. For example, an animated activity helps students to identify different colors. As part of this activity, different colors would appear randomly on the page.

Creating a website with animations is a complex task. Detailed knowledge of some scripting language is needed for this. Moreover, creating animations with scripts is a time-consuming process.

However, CSS provides an easy way to apply animation effects on HTML elements. It has a set of predefined properties and functions for applying transitions, animations, or transformations. Let us discuss the various CSS animation styles that can be applied on a Web page.

NOTE *CSS is supported in Webkit browsers, including Safari, Chrome, and Firefox. Webkit is a web browser engine that allows Web browsers to render Web pages that are not supported by them. As the technology is relatively new, prefixes for the browser vendors are required. The following keywords are needed to prefix with browser windows:*
Google Chrome -webkit-
Opera -o-

Applying Transitions

Consider an example where you want to change the color of the HTML element from red to black in the event of a mouse hover. This change is instantaneous. You cannot specify any time interval when the change should start or stop.

The CSS transitions provide a way to moderately change the HTML element from one position to another. With CSS enabled transitions, you can specify the duration of the transition property on which the transition effect will occur and the delay time for the transition. The following table displays the transition properties.

Property	*Definition*	*Valid Values*	*Sample Usage*
`transition-property`	*Is used to specify the name of the CSS property to which the transition is applied.*	`none|all| property name`	`{transition-property:width;}`
`transition-duration`	*Is used to specify the duration, a transition effect will take.*	`time (in seconds or milliseconds)`	`{transition-duration:2s;}`
`transition-timing-function`	*Is used to specify the speed curve of the transition effect.*	`linear|ease| ease-in|ease-out|ease-in-out`	`{transition-timing-function: linear}`
`transition-delay`	*Is used to specify the time duration for start of the transition effect.*	`time`	`{transition-delay:3s;}`
`transition`	*Is used to specify all the properties in one declaration.*	`property duration timing-function delay`	`{transition: width 2s;}`

The Transition Properties

NOTE *Internet Explorer does not support CSS transitions.*

Consider the following code to view the transition effect on the <DIV> element:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
div
{
width:200px;
height:200px;
background:red;
-webkit-transition:width 4s; /*
Chrome */
-o-transition:width 4s; /* Opera */
}
div:hover
{
width:400px;
}
</STYLE>
</HEAD>
<BODY>
<DIV></DIV>
</BODY>
</HTML>
```

In the preceding code, when the user points the mouse to the <DIV> element, the transition is applied on the <DIV> element to increase its width to 400 pixels in a time duration of four seconds.

The output derived by using the transition effect is

displayed, as shown in the following file.

The Output Derived by Using the Transition Effect

Applying Animations

CSS transitions provide an easy way to apply animation effects on HTML elements. For example, you can create an animation that will increase the size of the HTML element when the user clicks it. However, the transition states depend on its property values. The user does not have any control on the transition states. Therefore, the CSS animation comes into the existence.

A CSS animation provides a way to moderately change HTML elements from one position to another. With CSS enabled animations, you can specify how many times the animation iterates, whether it should alternate or not, and whether the animation should be in the running state or paused.

To specify the animation styles, you need to create certain animation rules known as `@keyframe`. It includes a set of properties and methods used to create an animation.

The following syntax can be used to create `@keyframe`:

```
@keyframe keyframename
{
from {property: value;}
to{property: value;}
}
```

In the preceding syntax:

- `keyframename`: Specifies the name of the keyframe.
- `from`: Specifies the initial style that you want to apply on the HTML element.
- `to`: Specifies the change that you want to reflect in the style.

However, if you want to embed more than one change in an animation, you can specify it by using percentage. Consider the following code to create a keyframe by using percentage:

```
<STYLE>
@-webkit-keyframes myfirst /* Chrome
*/
{
0% {background:red;}
20% {background:green; }
40% {background:yellow;}
80% {background:red;}
}
</STYLE>
```

The preceding code will create a keyframe named `myfirst`. The background color specified in percentage sets the background color to red when the animation starts, to green when the animation is 20% complete, to yellow when the animation is 40% complete, and to red again when the animation is 80% complete.

After creating the animation in keyframe, you need to connect it to the HTML element. Otherwise, the animation will have no effect. To connect the animation, you need to specify the animation properties. The following table lists the properties to define the name, duration, speed curve, and delay time for the animation.

Property	*Definition*	*Valid Values*	*Sample Usage*
`animation-name`	*Specifies the name of* `@keyframe` *animation.*	`keyframename|none`	`@keyframe first`
`animation-duration`	*Is used to set the time an animation will take to complete one cycle.*	`time`	`{animation-duration:2s;}`
`animation-timing-function`	*Is used to specify the speed curve of the animation.*	`linear|ease|ease-in|ease-out|ease-in-out`	`{animation-timing-function: linear;}`
`animation-delay`	*Is used to specify the time duration for start of the animation.*	`time`	`{animation-delay:4s;}`

The Animation Properties

The following table lists the properties to define the direction, iteration count, and play state for the animation.

Property	*Definition*	*Valid Values*	*Sample Usage*
`animation-direction`	*Is used to specify whether an animation should play in reverse on alternate cycles.*	`normal|alternate`	`{animation-direction:alternate;}`
`animation-iteration-count`	*Specifies how many times an animation should be played.*	`value|infinite`	`{animation-iteration-count:3;}` `{animation-iteration-count:infinite;}`
`animation-play-state`	*Is used to specify whether an animation is running or paused.*	`running|paused`	`{animation-play-state: paused;}`
`animation`	*Is used to set all the properties in one declaration, except the* `animation-play-state` *property.*	`name` `duration` `timing-function` `delay` `iteration-count` `direction`	`{animation mymove 5s 3;}` *The preceding code snippet sets* `duration` *of 5 seconds and* `iteration-count` *of 3 for the animation named mymove.*

The Animation Properties to Define Direction, Iteration Count, and Play State of Animation

Consider the following code in the `<STYLE>` tag to bind the animation to the `<DIV>` element:

```
div
{
width:100px;
height:100px;
```

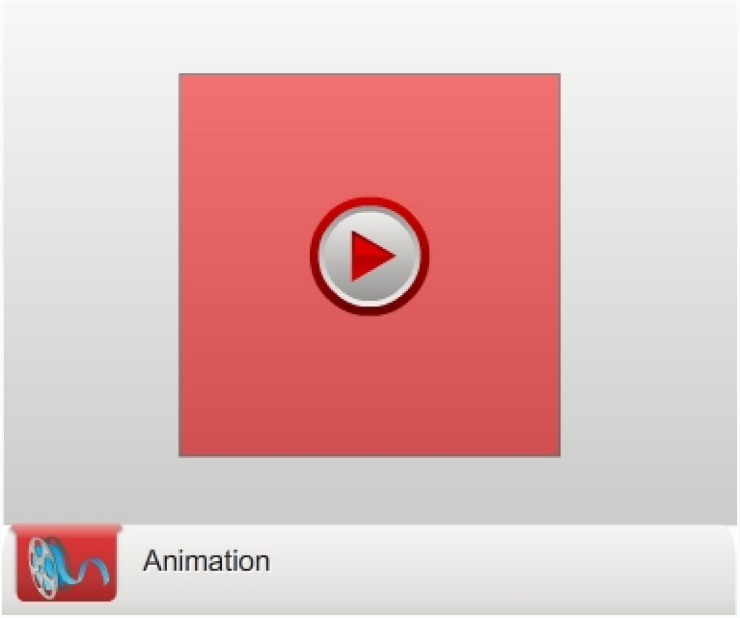
```
background:red;
position:relative;
-webkit-animation-name:myfirst; /*
Chrome */
-webkit-animation-duration:5s; /*
Chrome */
}
```

In the preceding code, the animation property is used in the<DIV> element to bind it to the keyframe,myfirst. The animation property takes two arguments. The first argument is the name of the keyframe and the second argument is the duration of the animation.

The following lines display the entire code for applying animation:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
div
{
width:100px;
height:100px;
background:red;
position:relative;
-webkit-animation-name:myfirst; /*
Chrome */
-webkit-animation-duration:5s; /*
Chrome */
}
@-webkit-keyframes myfirst /* Chrome
*/
{
0% {background:red;}
20% {background:green; }
40% {background:yellow;}
80% {background:red;}
}
</STYLE>
<DIV></DIV>
</BODY>
</HTML>
```

The output derived by applying animation effects is displayed, as shown in the following file.



The Output Derived by Applying Animation Effects

Internet Explorer does not support CSS animations.

Applying Transformations

Consider an example where you want to rotate a cube having different pictures on its curves. This way you want to add 2D or 3D effects on the Web page. CSS provides the functionality of transformations to modify the appearance of the HTML element. You can rotate, scale, and skew HTML elements easily by setting transformation properties. The CSS transformations have the following categories:

```
transform: none| transform-functions;
```

In the preceding syntax:

- `transform-functions`: Specifies the functions that can be used to transform elements.
- `none`: Specifies that there is no transformation. It is the default value that is set to the transform property.

CSS transformations have the following categories:

- 2D Transforms
- 3D Transforms

2D Transforms

With 2D transforms, you can apply various transformations, such as rotations or translations, on HTML elements. You can also apply more than one transformation on the HTML element. However, to apply transformations, you need to first understand the methods that can be used to transform HTML elements. The following table lists the 2D transform methods.

Function	***Syntax***	***Description***
`translate`	`translate (x,y);`	*Is used to move an element from its current position to the*

		position specified in the x-axis and y-axis.
`rotate`	`rotate (angle);`	*Is used to rotate an element at a given degree in the clockwise direction. You can also input a negative value, which will rotate the element in the counter clockwise direction.*
`skew`	`skew(x-angle, y-angle);`	*Is used to turn an element in a given angle, depending on the parameters given in the x-axis and y-axis.*
`scale`	`scale(x,y);`	*Is used to increase or decrease the size of an element, depending on the parameters given in the x-axis and y-axis.*
`matrix`	`matrix (n,n,n,n,n,n );`	*Is used to combine all the methods in one declaration.*

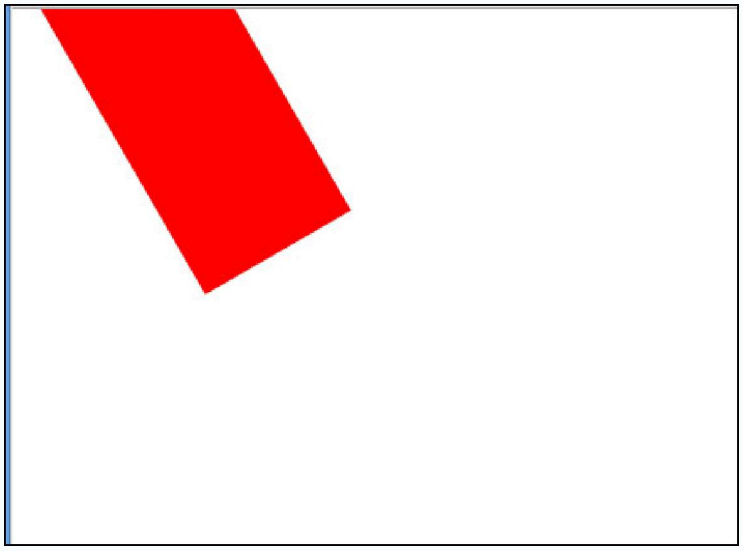
The 2D Transformation Methods

Consider the following code to rotate the HTML element:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
div
{
width:200px;
height:100px;
background-color:red;
transform:rotate(60deg);
-webkit-transform:rotate(60deg); /*
Chrome */
}
</STYLE>
</HEAD>
<BODY>
<DIV></DIV>
</BODY>
</HTML>
```

In the preceding code, the `<DIV>` element will get rotated to a 60 degree angle.

The output derived by applying transformations is displayed, as shown in the following figure.



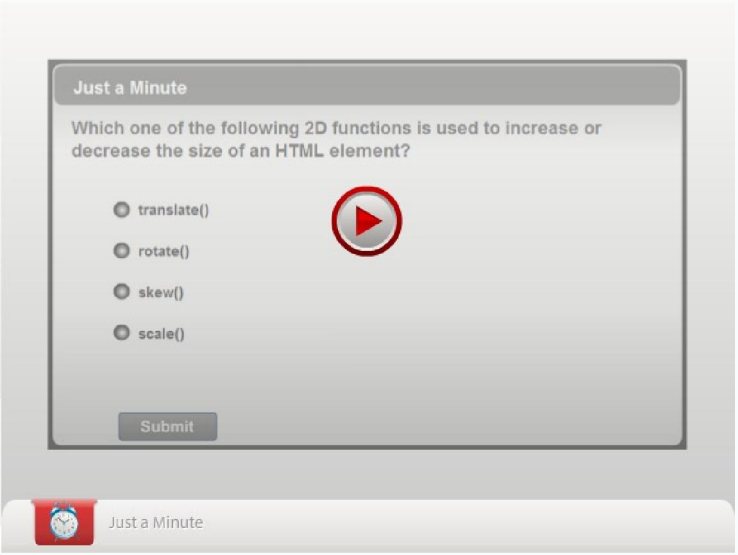
The Output Derived by Applying Transformations

3D Transforms

CSS also allows you to format the HTML element by using 3D transforms. The following transform methods can be used to apply 3D transforms:

- `rotateX()`: Rotates the element at a given degree around the x-axis.
- `rotateY()`: Rotates the element at a given degree around the y-axis.

3D methods are not supported in Internet Explorer and Opera.



Activity 2.2: Applying Transitions, Animations, and Transformations

Summary

In this chapter, you learned that:

- CSS is a collection of styles that allows you to change the appearance of HTML elements on Web pages.
- CSS allows the creation of one or more rules for defining the style characteristics of HTML elements. Each rule consists of the following parts:
  - Selector
  - Declaration Block
- To customize only few elements on a Web page, inline styles can be applied.
- Inline styles are attached directly with the tag in the HTML document. They are specified by using the `style` attribute inside an element declaration in the HTML document.
- An internal style sheet is used when there is a need to stylize the multiple occurrences of an element on a Web page with the same style.
- An external style sheet is used when multiple Web pages are to be styled in the same manner to ensure the consistent look and feel across the entire website.
- An ID selector is used to identify an element that you need to style differently from the rest of the page. An ID selector is defined by using the hash symbol (#).
- A CSS style can be applied to a group of elements by using the class selector. The class selector is defined by using a dot (.).
- A CSS property represents a characteristic of the HTML element that can be customized.
- The CSS properties can be divided into the following categories:
  - Font
  - Text
  - Link
  - List
  - Background
- The `display` property is used to set the appearance of an element on a Web page.
- The `visibility` property is used to specify whether an element should be visible or not.
- The `position` property is used to position an element on a Web page.
- The `float` property is used to place HTML elements to the left or right margin, in relation to the other HTML elements.
- The `clear` property is used to turn off the float effect on HTML elements.
- The CSS transitions provide a way to moderately change the HTML element from one position to another.
- To specify the animation styles, you need to create certain animation rules known as `@keyframe`, which is a rule, where the animation is created.
- CSS transformations have the following categories:
  - 2D Transforms
  - 3D Transforms

Reference Reading

Working with Styles

Reference Reading: Books	***Reference Reading: URLs***
CSS Cookbook	*http://www.w3schools.com/css/* *https://developer.mozilla.org/en-US/learn/css*

Applying Transitions, Animations, and Transformations

Reference Reading: Books	***Reference Reading: URLs***
The Definitive Guide to HTML5 By Adam Freeman	*https://developer.mozilla.org/en-US/docs/CSS/transform* *http://www.the-art-of-web.com/css/css-animation/#.UT772xzfDts*

Chapter 3

Working with Tables and Frames

As you keep adding elements to a page one after the other, the page gets cluttered. As a result, it becomes difficult for the user to read and understand the information given on the Web page. To resolve this issue, the information needs to be arranged in a logical and presentable manner. For this, you can use a table to design your Web pages.

Tables are used for structuring and displaying complex information in a structured format on a Web page. However, as the information grows in complexity, there is a need to segregate and display information in different sections of the Web page.

Frames provide a mechanism for positioning and displaying several Web pages in different sections of a single browser window.

This chapter discusses the need and usage of tables and frames. In addition, it discusses the attributes that can help customize the Web page created by using frames and tables.

Objectives

In this chapter, you will learn to:

- Create tables
- Access multiple Web pages by using frames

Creating Tables

Consider a scenario of online tutorial of LearnMySQL website that teaches how to access and manipulate data by using MySQL. In addition, it teaches the concepts of database administration by using MySQL. One of the Web pages in this tutorial displays the content on data types in a format, as shown in the following figure.

- Char(M) is a Fixed-Length string between 0 to 255 characters.
- Varchar(M) is a Variable-Length string between 0 to 255 characters.

The Content on Data Types

In the preceding figure, the content on data types can be made more presentable and understandable by arranging the same in the form of a table, as shown in the following figure.

Name	Length	Range
Char(M)	Fixed	0-255
Varchar(M)	Variable	0-255

The Content Displayed in the Form of a Table

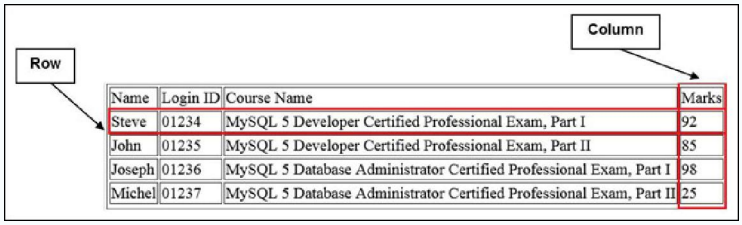
The information on a Web page should be well structured and organized to ensure that it is readable. Tables are used in Web pages to enhance the readability by presenting information in a structured way.

Tables represent complex information in a simple and easy-to-understand manner. By using tables, information can be arranged logically in rows and columns. You can also control the spacing, layout, and presentation of tabular data on a Web page.

Identifying the Basic Structure of a Table

The LearnMySQL website provides tutorials to the users for preparing different certification exams of MySQL. Once registered for an online exam, a user is provided a login ID and password. The test results of the candidates, which comprise information such as name, login ID, course name, and marks, are thereafter displayed on the website.

Tables can be used to display such information, which are arranged in the form of horizontal rows and vertical columns, as shown in the following figure.



Name	Login ID	Course Name	Marks
Steve	01234	MySQL 5 Developer Certified Professional Exam, Part I	92
John	01235	MySQL 5 Developer Certified Professional Exam, Part II	85
Joseph	01236	MySQL 5 Database Administrator Certified Professional Exam, Part I	98
Michel	01237	MySQL 5 Database Administrator Certified Professional Exam, Part II	25

The Test Results Table

In the preceding table, each row displays the test details of a candidate. You can create such a table in HTML by using the `<TABLE>` tag. The `<TABLE>` tag acts as a container for tags that are used for creating rows and columns and adding data. A table contains an opening `<TABLE>` tag and a closing `</TABLE>` tag. In addition, you can also control the appearance of the border around a table. This is done by using the `border` attribute of the `<TABLE>` tag. If used with the `<TABLE>` tag, the `border` attribute applies a border around the table and can be assigned the value, `1`, or "". If you do not want a border to appear around the table, do not use the `border` attribute with the `<TABLE>` tag.

The table is divided into the following logical sections:

- Body
- Header
- Footer

Specifying Table Body

The table body contains data arranged in rows. Each row further comprises one or multiple columns. The rows of the body of a table can be grouped by using the `<TBODY>` tag.

You can use more than one `<TBODY>` tag inside the `<TABLE>` tag. This is done when you want to logically group the rows of the table body for applying different presentation styles to each group.

Creating Rows

For adding rows to a table, the <TR> tag is used. The contents of a row are placed between the <TR> and </TR> container tags. The content of each row comprises one or more column values. The number of rows in a table depends on the number of <TR> tags within the <TABLE> tag.

Creating Columns

For adding columns to a row, the <TD> tag is used. The <TD> tag specifies the content of the columns. The content is enclosed within the <TD> and </TD> container tags. The <TD> tags are used within the <TR> and </TR> tags.

Consider the following code snippet that shows the use of the <TR> and <TD> tags in HTML:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<TABLE border = "1">
<TBODY>
<TR><TD> Name </TD>
<TD> Login ID</TD>
<TD> Course Name</TD>
<TD> Marks</TD></TR>
<TR>
<TD>Steve</TD>
<TD>01234</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part I</TD>
<TD>92</TD>
</TR>
<TR>
<TD>John</TD>
<TD>01235</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part II</TD>
<TD>85</TD>
</TR>
<TR>
<TD>Joseph</TD>
<TD>01236</TD>
<TD>MySQL 5 Database Administrator
Certified Professional Exam, Part I
</TD>
<TD>98</TD>
</TR>
<TR>
<TD>Michel</TD>
<TD>01237</TD>
<TD>MySQL 5 Database Administrator
Certified Professional Exam, Part
II</TD>
<TD>25</TD>
</TR>
<TBODY>
</TABLE>
</BODY>
</HTML>
```

In the preceding code snippet, a table with five rows and four columns is created. The first row of the table specifies the headings for individual columns as Name, Login ID, Course Name, and Marks; and the rest four rows are populated as the different values of the headings. The preceding code snippet creates the Test Results table, as shown in the following figure.

Name	Login ID	Course Name	Marks
Steve	01234	MySQL 5 Developer Certified Professional Exam, Part I	92
John	01235	MySQL 5 Developer Certified Professional Exam, Part II	85
Joseph	01236	MySQL 5 Database Administrator Certified Professional Exam, Part I	98
Michel	01237	MySQL 5 Database Administrator Certified Professional Exam, Part II	25

The Test Results Table

Combining Multiple Rows and Columns into a Single Cell

Consider the scenario of an online tutorial of LearnMySQL website. On this website, you want to display the result of the top three performers in a tabular form. The table should appear in such a way that only the column of the first row should extend to four columns and display the heading, **Top Three Performers**. The remaining rows of the table should have four separate columns. To extend or merge the cells up to the desired columns, you can use the colspan attribute. Similarly, to merge the rows, you can use the rowspan attribute.

The colspan attribute is specified within the table cell to define the number of columns the cell can extend to. The default value for colspan is 1. The following code snippet uses the colspan attribute to extend a cell up to four columns in the first row of the table:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<TABLE border = "1">
<TBODY>
<TR><TD colspan= "4"><CENTER>Top
Three Performers</CENTER></TD></TR>
<TR><TD> Name </TD>
<TD> Login ID</TD>
<TD> Course Name</TD>
<TD> Marks</TD></TR>
<TR>
<TD>Joseph</TD>
<TD>01236</TD>
<TD>MySQL 5 Database Administrator
Certified Professional Exam, Part I
</TD>
<TD>98</TD>
</TR>
<TR>
<TD>Steve</TD>
<TD>01234</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part I</TD>
<TD>92</TD>
</TR>
```

```
<TR>
<TD>John</TD>
<TD>01235</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part II</TD>
<TD>85</TD>
</TR>
</TBODY>
</TABLE>
</BODY>
</HTML>
```

In the preceding code snippet, the `Top Three Performers` column extends horizontally, acquiring the space of four columns to appear as a single column. The output derived after using the `colspan` attribute is displayed in the following figure.

Top Three Performers			
Name	Login ID	Course Name	Marks
Joseph	01236	MySQL 5 Database Administrator Certified Professional Exam, Part I	98
Steve	01234	MySQL 5 Developer Certified Professional Exam, Part I	92
John	01235	MySQL 5 Developer Certified Professional Exam, Part II	85

The Output Derived After Using the Colspan Attribute

The `rowspan` attribute is specified within table cell to define the number of rows a cell can extend to. The default value for `rowspan` is `1`. The following code snippet uses the `rowspan` attribute to extend a cell up to five rows in the first column of the table:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<TABLE border = "1">
<TBODY>
<TR><TD rowspan= "5">Top Three
Performers</TD></TR>
<TR><TD> Name </TD>
<TD> Login ID</TD>
<TD> Course Name</TD>
<TD> Marks</TD></TR>
<TR>
<TD>Joseph</TD>
<TD>01236</TD>
<TD>MySQL 5 Database Administrator
Certified Professional Exam, Part I
</TD>
<TD>98</TD>
</TR>
<TR>
<TD>Steve</TD>
<TD>01234</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part I</TD>
<TD>92</TD>
</TR>
<TR>
<TD>John</TD>
<TD>01235</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part II</TD>
<TD>85</TD>
</TR>
</TBODY>
</TABLE>
</BODY>
</HTML>
```

In the preceding code snippet, the `Top Three Performers` row extends vertically acquiring the space of five rows to appear as a single row. The output derived after using the `rowspan` attribute is displayed, as shown in the following figure.

Top Three Performers	Name	Login ID	Course Name	Marks
	Joseph	01236	MySQL 5 Database Administrator Certified Professional Exam, Part I	98
	Steve	01234	MySQL 5 Developer Certified Professional Exam, Part I	92
	John	01235	MySQL 5 Developer Certified Professional Exam, Part II	85

The Output Derived After Using the Rowspan Attribute

Specifying Table Header

The table header is a row that contains the headings for the columns of the table. The headings in the table provide a brief explanation about the data in the subsequent rows of the table. For example, if the heading of a column is User ID, it specifies that the column displays the users' ID in the following rows. In addition, you can group the header content of a table so that it stands out from the rest of the rows by using the `<THEAD>` tag. This tag specifies that a group of rows are the header rows.

Creating Heading for the Table Columns

To create the headings for the table columns, you can use the `<TH>` tag. The `<TH>` tag displays the heading in bold and is centrally aligned.

The following code snippet shows the use of the `<TH>` tag:

```
<TABLE border = "1">
<THEAD>
<TR>
<TH> Name </TH>
<TH> Login ID</TH>
<TH> Course Name</TH>
<TH> Marks</TH>
</TR>
</THEAD>
.........
</TABLE>
```

In the preceding code snippet, `Name`, `Login ID`, `Course Name`, and `Marks` represent the table header columns.

The preceding code snippet creates a table header in the first row, as shown in the following figure.

Name	Login ID	Course Name	Marks
Steve	01234	MySQL 5 Developer Certified Professional Exam, Part I	92
John	01235	MySQL 5 Developer Certified Professional Exam, Part II	85
Joseph	01236	MySQL 5 Database Administrator Certified Professional Exam, Part I	98
Michel	01237	MySQL 5 Database Administrator Certified Professional Exam, Part II	25

A Table Header

Specifying Table Footer

Consider a situation where you need to provide the summary for table rows. For example, at the end of the table, you need to specify the total number of students. For this, you can use the <TFOOT> tag, which is always specified after the <THEAD> tag but before the <TBODY> tag. The <TFOOT> tag should contain at least one <TR> tag.

Consider the following code snippet that shows the use of the <TFOOT> tag in HTML:

```
<TABLE border = "1">
<THEAD>
<TR><TH colspan= "4">Top Three
Performers</TH></TR>
<TR>
<TH> Name </TH>
<TH> Login ID</TH>
<TH> Course Name</TH>
<TH> Marks</TH>
</TR>
</THEAD>
<TFOOT>
<TR><TD colspan= "3"> Total number of
students </TD>
<TD> 3</TD></TR>
</TFOOT>
<TBODY>
<TR >
<TD>Joseph</TD>
<TD>01236</TD>
<TD>MySQL 5 Database Administrator
Certified Professional Exam, Part I
</TD>
<TD>98</TD>
</TR>
<TR>
<TD>Steve</TD>
<TD>01234</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part I</TD>
<TD>92</TD>
</TR>
<TR>
<TD>John</TD>
<TD>01235</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part II</TD>
<TD>85</TD>
</TR>
</TBODY>
</TABLE>
```

The preceding code snippet creates the footer for the table, as shown in the following figure.

Top Three Performers			
Name	**Login ID**	**Course Name**	**Marks**
Joseph	01236	MySQL 5 Database Administrator Certified Professional Exam, Part I	98
Steve	01234	MySQL 5 Developer Certified Professional Exam, Part I	92
John	01235	MySQL 5 Developer Certified Professional Exam, Part II	85
Total number of students			3

The Output Displaying the Table Footer

Specifying the Table Caption

After creating a table, you should define a title for it. A caption specifies a title for the table. It enables the users to understand the information that the table displays.

For creating a table title, the <CAPTION> tag is used. Once specified, it becomes the first element after the <TABLE> tag. It has an opening tag, <CAPTION>, and a closing tag, </CAPTION>. These tags are used within the <TABLE> tags.

Consider the following code snippet that shows the use of the <CAPTION> tag in HTML:

```
<TABLE border = "1">
<CAPTION> Test Results </CAPTION>
<THEAD>
<TR><TH colspan= "4">Top Three
Performers</TH></TR>
<TR>
<TH> Name </TH>
<TH> Login ID</TH>
<TH> Course Name</TH>
<TH> Marks</TH>
</TR>
</THEAD>
<TFOOT>
<TR><TD colspan= "3"> Total number of
students </TD>
<TD> 3</TD></TR>
</TFOOT>
<TBODY>
<TR >
<TD>Joseph</TD>
<TD>01236</TD>
<TD>MySQL 5 Database Administrator
Certified Professional Exam, Part I
</TD>
<TD>98</TD>
</TR>
<TR>
<TD>Steve</TD>
<TD>01234</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part I</TD>
<TD>92</TD>
</TR>
<TR>
<TD>John</TD>
<TD>01235</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part II</TD>
<TD>85</TD>
</TR>
```

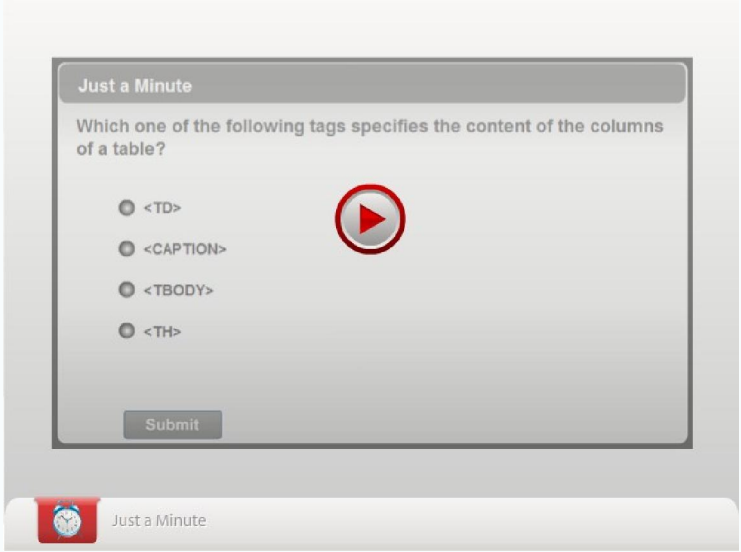
```
</TBODY>
</TABLE>
```

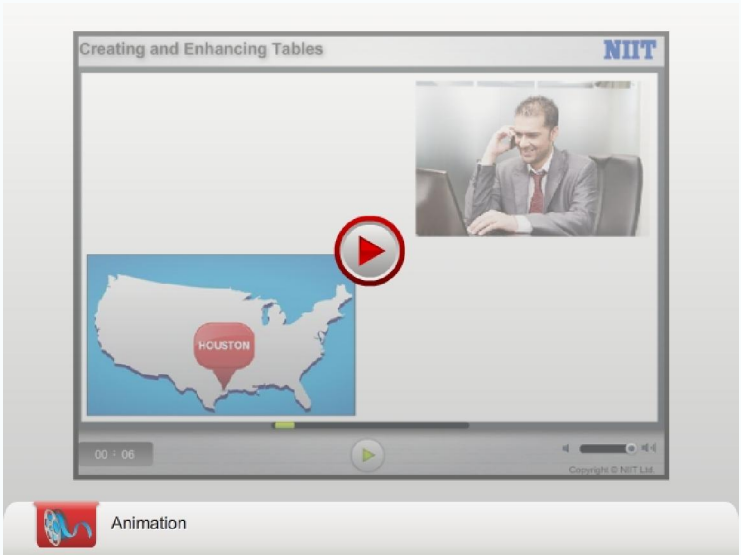
The preceding code snippet assigns a caption to the Test Results table, as shown in the following figure.

Test Results

Top Three Performers			
Name	**Login ID**	**Course Name**	**Marks**
Joseph	01236	MySQL 5 Database Administrator Certified Professional Exam, Part I	98
Steve	01234	MySQL 5 Developer Certified Professional Exam, Part I	92
John	01235	MySQL 5 Developer Certified Professional Exam, Part II	85
Total number of students			3

The Output Displaying the Table Caption





Enhancing Tables

The Test Results table displays the information in an understandable and presentable manner. However, it can always be enhanced to have a better look and feel. For example, the border of a particular thickness and color can be applied on the table. In addition, the width and the height of the table can also be set. To apply such effects on a table for enhancing its appearance, you need to use CSS.

CSS provides various properties to enhance the visual appearance of a table. The following table lists the various CSS properties that can be used to stylize the table to be displayed on a Web page.

Name	***Description***	***Example***
`border`	*It is used to set the style, width, and color of the table border. It is used with* `<TABLE>`, `<TH>`, *and* `<TD>` *tags.*	`table { border:dotted 1px brown; }` *The preceding code snippet sets the table border to 1 pixel with dotted border line in brown color.*
`border-collapse`	*It is used to set whether the table borders are collapsed into a single border or they appear separated. By default, a double border appears around a table because both the table and the columns have separate borders. The possible values for the* `border-collapse` *property are* `collapse` *and* `separate`.	`table { border-collapse:collapse; }` *The preceding code snippet collapses the table borders into a single border.* `table { border-collapse:separate; }` *The preceding code snippet keeps the table borders separated.*
`height and width`	*These are used to define the height and width of a table. It is used with* `<TABLE>`, `<TH>`, *and* `<TD>` *tags.*	`table { width:100px; height:50px; }` *The preceding code snippet sets the width and height of the border to 100px and 50px, respectively.*
`text-align`	*It is used to align the text in a table. Its values are left, right, and center. It is used with* `<TABLE>`, `<TH>`, *and* `<TD>` *tags.*	`table { text-align:right; }` *The preceding code snippet right aligns the text.*

padding	It is used to control the space between the table and the content in the table. It is used with <TABLE>, <TH>, and <TD> tags.	td
		{
		padding:10px
		;
		}
		The preceding code snippet sets the space between the border and the content of the table to 10px.

color	It is used to set the color of the border and text in a table. It is used with the <TABLE>, <TH>, and <TD> tags.	table
		{
		color:blue;
		}
		The preceding code snippet sets the color of the border and text to blue.

background-color	It is used to set the background color of the entire table or columns. It is used with the <TABLE>, <TH>, and <TD> tags.	th
		{
		background-color:gray;
		}
		The preceding code snippet sets the background color of the table header to gray.

The CSS Properties to Stylize a Table

Consider the following code to enhance the look and feel of the Test Results table:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<STYLE>
th
{
font-size:18px;
text-align:center;
padding-top:3px;
background-color:#BDB76B;
color:#006400;
}
caption
{
font-size:35px;
color:Black;
}
td
{
font-size:15px;
border:1px solid #008B8B;
padding:3px ;
background-color: Khaki;
}
tr.alt td
{
color:#F0E68C;
background-color:#B22222;
}
</STYLE>
</HEAD>
<BODY>
<TABLE border = "1">
<CAPTION> Test Results </CAPTION>
<THEAD>
<TR><TH colspan= "4">Top Three
Performers</TH></TR>
<TR>
<TH> Name </TH>
<TH> Login ID</TH>
<TH> Course Name</TH>
<TH> Marks</TH>
</TR>
</THEAD>
<TFOOT>
<TR ><TD colspan= "3"> Total number
of students </TD>
<TD> 3</TD></TR>
</TFOOT>
<TBODY>
<TR class = "alt">
<TD>Joseph</TD>
<TD>01236</TD>
<TD>MySQL 5 Database Administrator
Certified Professional Exam, Part I
</TD>
<TD>98</TD>
</TR>
<TR>
<TD>Steve</TD>
<TD>01234</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part I</TD>
<TD>92</TD>
</TR>
<TR class = "alt">
<TD>John</TD>
<TD>01235</TD>
<TD>MySQL 5 Developer Certified
Professional Exam, Part II</TD>
<TD>85</TD>
</TR>
</TBODY>
</TABLE>
</BODY>
</HTML>
```

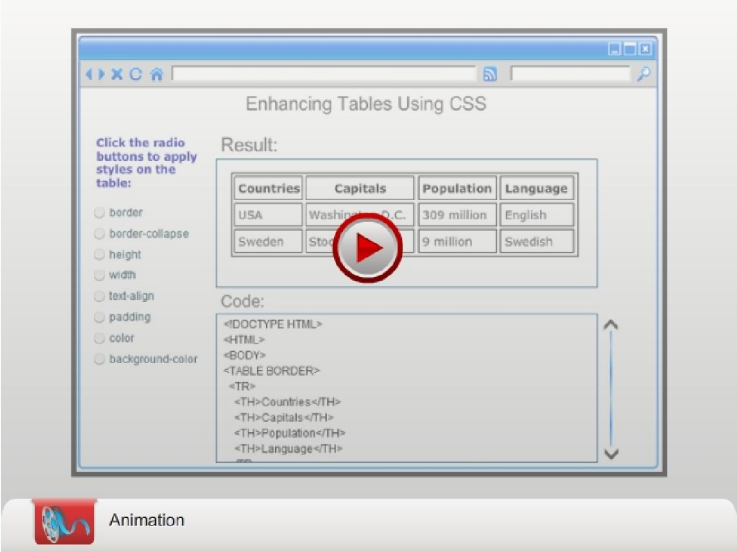
In the preceding code, different CSS rules have been created to stylize the table caption, heading, and data cells. The `th` selector sets the font, color, and alignment of the table header. The `caption` selector sets the font and color of the table caption. The`td` selector sets the size and color of the table data. The `tr.alt td` selector is created to apply styles to the rows having class selector, `alt`.

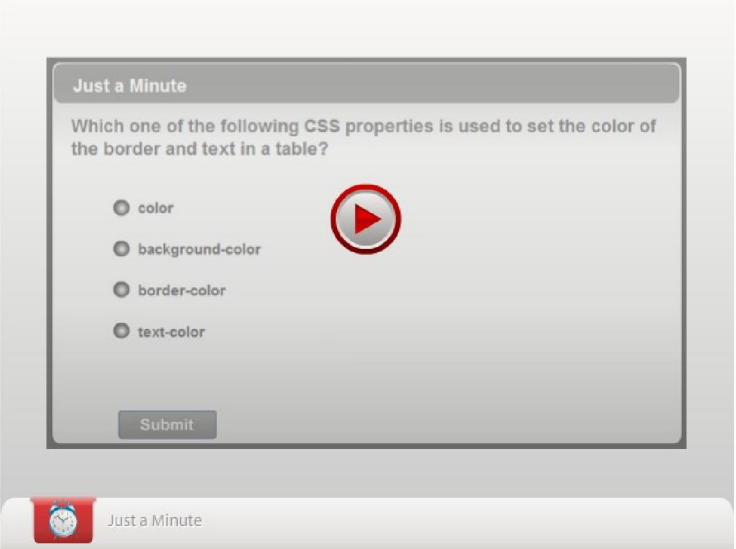
The output of the preceding code is displayed in the following figure.

Test Results

Top Three Performers			
Name	Login ID	Course Name	Marks
Joseph	01236	MySQL 5 Database Administrator Certified Professional Exam, Part I	98
Steve	01234	MySQL 5 Developer Certified Professional Exam, Part I	92
John	01235	MySQL 5 Developer Certified Professional Exam, Part II	85
Total number of students			3

The Table Obtained After Applying CSS Styles

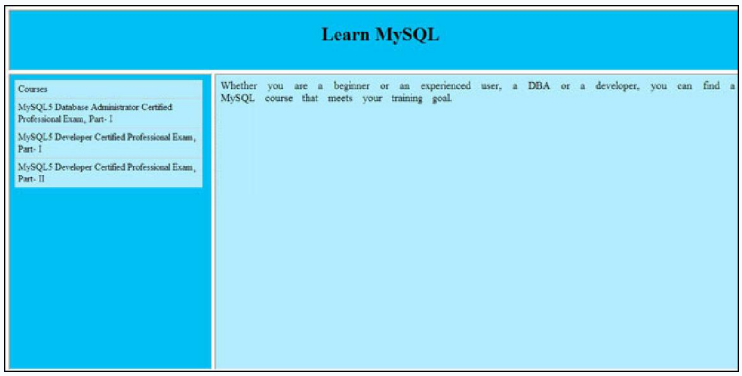




Activity 3.1: Creating a Table

Accessing Multiple Web Pages Using Frames

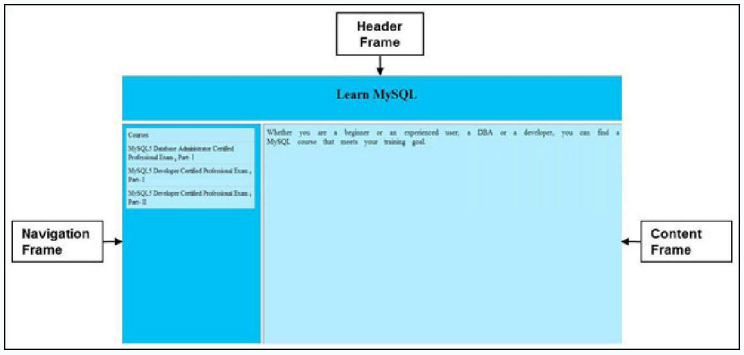
The LearnMySQL website provides the learning content on the courses that are mapped with different online certification exams of MySQL. The following figure displays a Web page from the website.



A Web Page from the LearnMySQL Website

Although the page in the preceding figure looks similar to an ordinary Web page, it comprises three separate Web pages. These Web pages are displayed in the same browser window, which makes it look like a single page. This is made possible by using the concept of frames. A frame is a rectangular region in a browser window inside which a Web page can be displayed.

On the Web page of LearnMySQL website, the browser window is divided into three rectangular regions by using three frames, as shown in the following figure.



The Browser Window Divided into Frames

In the preceding figure, the first frame that is named as HeaderFrame, displays the name of the website. The second frame that is named as NavigationFrame, contains the links to the various MySQL courses. The third frame that is named as ContentFrame, displays the Web page corresponding to the link that the user clicks in the navigational links section.

Creating Web Pages Using Frames

In the LearnMySQL website, the table of contents is displayed in the left pane. When a user clicks a link, the corresponding course is displayed in the right pane. This functionality can be achieved by using frames.

Before creating a Web page that has frames, you need to decide the Web pages you want to display in the frames and the structural appearance of these Web pages in the resulting Web page. Then, you must create a Web page that is displayed in each frame. Further, you need to organize these Web pages by putting them together inside a single browser window by using frames that can be implemented by using the `<IFRAME>` tag.

Exploring the <IFRAME> Tag

The HTML `<IFRAME>` tag is used to specify an inline frame. It allows you to divide a Web page into sections or frames. Each section can be used to display an individual Web page. Therefore, the `<IFRAME>` tag is used to embed an HTML Web page within another Web page. The embedded Web page is said to be contained within the other Web page, which is known as the containing page. The following attributes can be used with the `<IFRAME>` tag:

- `src`: Is used to specify the location or the URL of the Web page to be embedded inside the frame.
- `name`: Is used to assign a name to the frame.
- `seamless`: Is a boolean attribute, which instructs the browser to display the frame as a part of the containing Web page. If this attribute is used, the frame is displayed without scroll bars and border.
- `srcdoc`: Is used to specify an HTML code that defines the content to be displayed inside the frame.
- `height`: Is used to set the height of the frame.
- `width`: Is used to set the width of the frame.

As per the preceding figure, consider the following code to divide the Web page in three frames in the **home.html** page of the LearnMySQL website:

```
<!DOCTYPE HTML><HTML>
<BODY>
<IFRAME name="HeaderFrame"
width="91%" height="100" ></IFRAME>
<IFRAME name="NavigationFrame"
width="25%" height="500"></IFRAME>
<IFRAME name="ContentFrame"
width="65%" height="500" ></IFRAME>
</BODY>
</HTML>
```

In the preceding code, the `<IFRAME>` tag is used to divide the Web page, **home.html**, into frames of different width and height. Now, to make the Web pages visible inside the frames, you need to create, and then embed these Web pages inside the frames.

For example, the navigational links for the website are created inside a file named **nav.html** by using the following code:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
body{
background-color: #00C0F3;
}
            ul {
          margin: 0;
          padding: 0;
          list-style: none;// remove
bullets from the unordered list
          width: 350px;
          }



        ul li a {
          display: block;
          text-decoration: none;
          color: Black;
          background: #B3ECFC;
          padding: 5px;
          border: 1px solid #ccc;
          border-bottom: 0;
          }
        ul {
          margin: 0;
          padding: 0;
          list-style: none;
          width: 320px;



          }
        li:hover ul { display:
block; }
          a:hover
```

```
        {
                background-
color:#5E9DC9;
                                }


     </STYLE>



</HEAD>
<BODY>
<UL>
<LI><A href="Courses.html">Courses</
A></LI>
<LI><A href="Book1.html">MySQL5
Database Administrator Certified
Professional Exam Part- I</A></LI>
<LI><A href="Book2.html">MySQL5
Developer Certified Professional Exam
Part- I</A></LI>
<LI><A href="Book3.html">MySQL5
Developer Certified Professional Exam
Part- II</A></LI>
   </UL>
</BODY></HTML>
```

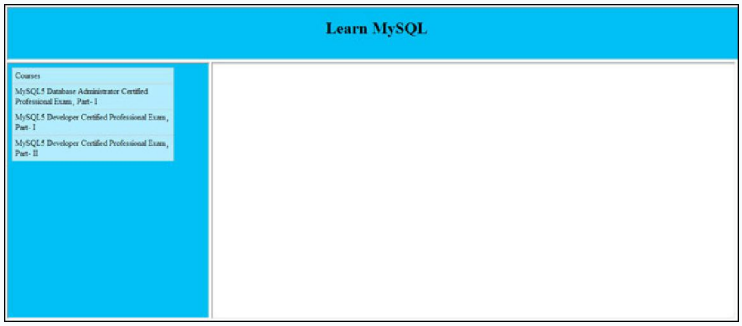
Similarly, other Web pages can also be created by using the HTML code.

You need to replace the code in the **home.html** Web page with the following code to embed the files inside the desired frames:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<IFRAME srcdoc= "<CENTER><H1><B>Learn
MySQL</B></H1></CENTER> "
name="HeaderFrame" width="91%"
height="100" ></IFRAME>
<IFRAME src="nav.html"
name="NavigationFrame" width="25%"
height="500"></IFRAME>
<IFRAME name="ContentFrame"
width="65%" height="500" ></IFRAME>
</BODY>
</HTML>
```

In the preceding code, the file, **header.html**, is embedded inside the frame,**HeaderFrame**. Similarly, the file, **nav.html**, is embedded in the frame named **NavigationFrame**. This is done by providing the name of the files as a value for the `src` attribute of the corresponding`<IFRAME>` tags used to create the frames. However, the content frame is not displaying anything as no HTML page has been specified for it.

The **home.html** Web page of the LearnMySQL website is displayed, as shown in the following figure.

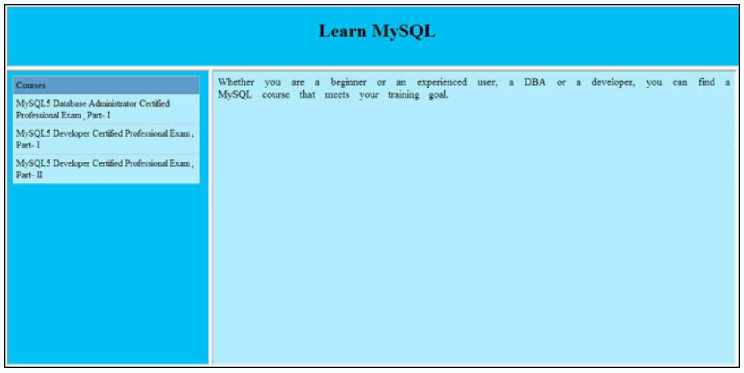


The Home Page of the LearnMySQL Website

In the preceding figure, the navigational links are displayed inside the frame, **NavigationFrame**. However, whenever a link is clicked, the corresponding Web page is displayed in a new window. Instead, you want to display these pages inside the frame, **ContentFrame**. This can be implemented by specifying the target frame for the links. You can specify a target frame for the links by using the `target` attribute of the `<A>` tag. The `target` attribute is used to specify the name of the frame where the HTML document should open. Consider the following code snippet to apply the `target` attribute on the links created in**nav.html**:

```
<!DOCTYPE HTML><HTML>
<BODY>
<UL>
<LI><A href="Courses.html"
target="ContentFrame">Courses</A></
LI>
<LI><A href="Book1.html"
target="ContentFrame">MySQL5 Database
Administrator Certified Professional
Exam Part- I</A></LI>
<LI><A href="Book2.html"
target="ContentFrame">MySQL5
Developer Certified Professional Exam
Part- I</A></LI>
<LI><A href="Book3.html"
target="ContentFrame">MySQL5
Developer Certified Professional Exam
Part- II</A></LI>
</UL>
</BODY>
</HTML>
```

In the preceding code snippet, the target attribute is used to specify **ContentFrame** as a target frame for the links. On clicking a link, the corresponding Web page is displayed inside the frame, **ContentFrame**, as shown in the following figure.



The Web Page Displayed Inside ContentFrame

Styling Frames

Frames can also be enhanced for a better look and feel. For example, the border of a particular thickness or appropriate margins can be applied on the frames to improve its appearance. This can be implemented by using the styling rules of CSS. The following table lists the various CSS properties that can be used for styling the frames.

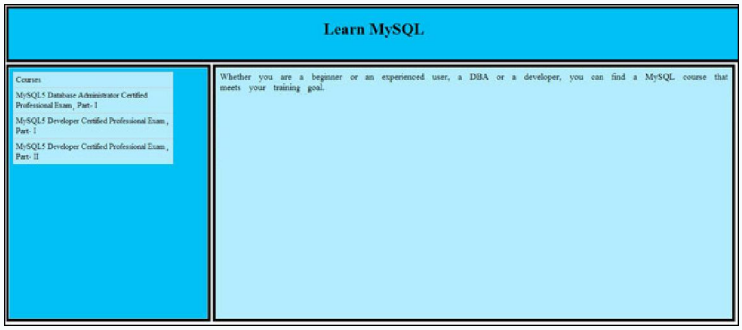
Name	***Description***	***Example***
`margin`	*It specifies an area around an element. It does not have a background color.*	`iframe { margin: 2px; }` *The preceding code snippet sets the margin of the frame to 2px.*
`padding`	*It defines the space between the element border and the element content.*	`iframe { padding: 2px; }` *The preceding code snippet sets the space between the border and the content of the frame to 2px.*
`border`	*It is used to apply a border of specific thickness and color on the frame. Its values are* `width`*,* `color`*, and* `style`*.*	`iframe { border: 2px solid black; }` *The preceding code snippet applies a border of 2px thickness and solid black color to the frame.*
`scrolling`	*It controls the appearance of a scrollbar around the frames. The possible values for this property are* `yes`*,* `no`*, and* `auto`*.*	`iframe { scrolling: auto; }` *The preceding code snippet applies a scrollbar on the frame, if required.*

The CSS Properties for Styling a Frame

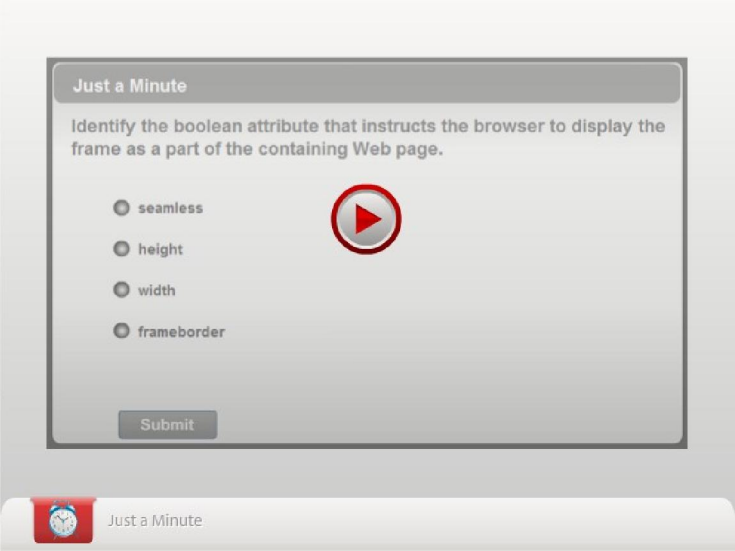
Consider the following code snippet for applying styles on the frames created in the file, **home.html**:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<STYLE>
iframe{
border :5px solid black;
margin-left :1px;
padding: 1px;
}
</STYLE>
</HEAD>
.......
</HTML>
```

The preceding code snippet applies a black color border to the frames with the left margin of 1px. The output of the preceding code snippet is displayed in the following figure.



The Code Output Displaying Frames in a Web Page After Applying CSS



Activity 3.2: Accessing Multiple Web Pages Using Frames

Summary

In this chapter, you learned that:

- Tables are used in Web pages to enhance the readability by presenting information in a structured way.
- You can create a table in HTML by using the `<TABLE>` tag.
- The rows of the body of a table can be grouped by using the `<TBODY>` tag.
- For adding rows to a table, the `<TR>` tag is used.
- For adding columns to a row, the `<TD>` tag is used.
- To extend or merge the cells up to the desired columns, you can use the `colspan` attribute. Similarly, to merge the rows, you can use the`rowspan` attribute.
- The table header is a row that contains the headings for the columns of the table.
- To create the headings for the table columns, you can use the `<TH>` tag.
- For creating a table title, the `<CAPTION>` tag is used.
- The HTML `<IFRAME>` tag is used to specify an inline frame. It allows you to divide a Web page into sections or frames.

Reference Reading

Creating Tables

Reference Reading: Books	***Reference Reading: URLs***
Introducing HTML5 By Bruce Lawson, Remy Sharp	*http://www.w3.org/TR/html401/struct/tables.html* *http://www.htmlquick.com/tutorials/tables.html*

Accessing Multiple Web Pages Using Frames

Reference Reading: Books	***Reference Reading: URLs***
The Definitive Guide to HTML5 By Adam Freeman	*http://www.w3schools.com/html/html_iframe.asp*

Chapter 4

Adding Interactivity to Web Pages

A Web page may require users to enter some input and generate output. For example, you may require creating a Web page for a shopping website that enables users to select the product they want to buy from a list and enter its quantity in a text box. The moment users specify these values, the total payable amount should be generated in another text box. This type of functionality cannot be performed by HTML alone. Therefore, a scripting language is required to add such functionality to the Web page.

This chapter discusses the basics of a scripting language. It explains the process of implementing JavaScript in Web pages. In addition, it discusses the usage of expressions and control structures. Moreover, it explains the need and use of functions.

Objectives

In this chapter, you will learn to:

- Understand scripting
- Implement JavaScript in Web pages
- Use variables, operators, and control structures
- Implement functions

Understanding Scripting

Consider a scenario where you need to create a Web page for a website called OnlineShop.com. The website facilitates users to register by using an online registration form. At the time of registration, the users are asked to select either of the two options, email or Mail, as the mode of communication. Upon selecting the email option, the user is provided a text field to enter the email address. However, if the user selects the Mail option, a text area is provided to enter the postal mailing address. The users are provided the statements of their transactions, acknowledgement receipts, or promotional offer mailers at the specified address. Such Web pages can be developed by using a scripting language.

Controls, such as text boxes, radio buttons, and text areas, will be discussed in the next chapter.

Types of Scripting

To create a dynamic and interactive Web page, you need to incorporate a block of code, which is known as script, in the Web page. The script can be executed either by the Web browser or by the Web server.

When a user requests for a Web page through a Web browser, the request is sent to a computer that is placed at a different location on the World Wide Web (WWW). The computer on which the browser is running is known as the client, and the computer that receives the request is known as the Web server.

When the Web server receives the request, it processes the request and sends the requested Web page to the client, which is, then, displayed in the browser window of the client.

Consider the example of the website of a famous fortnightly science journal. It allows its customers to subscribe online for the journal. While subscribing, the user needs to provide the contact details in the form of the permanent address and communication address. The communication address of the user can be the same as the permanent address. After filling the permanent address details, if the user selects the check box titled Same as Communication Address, the same address should be filled in the Communication Address field. This functionality can be implemented by using the script that is interpreted at the client-end itself.

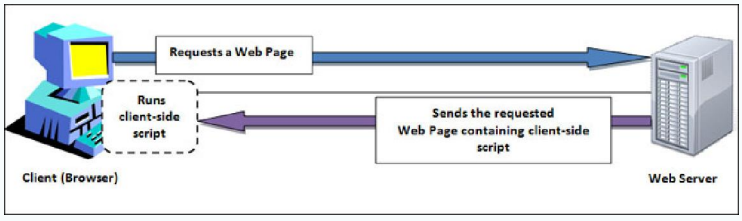
Scripting can be of two types. These are:

- Client-side scripting
- Server-side scripting

Client-side Scripting

Client-side scripting refers to the scripts that are executed at the client-side by the Web browser, running on the user's computer. Some of the languages used for creating client-side scripts are client-side JavaScript (CSJS) and Visual Basic Script (VBScript). When the Web browser requests a Web page, the server sends the requested Web page, which includes both, the HTML statement and the script statement, over the network. The Web browser reads the Web page and displays the results generated by interpreting the HTML statements. In addition, the Web browser executes the script statements as and when they are encountered while rendering the Web page.

The communication between a client and a server in case of client-side scripting is shown in the following figure.



The Communication in Client-side Scripting

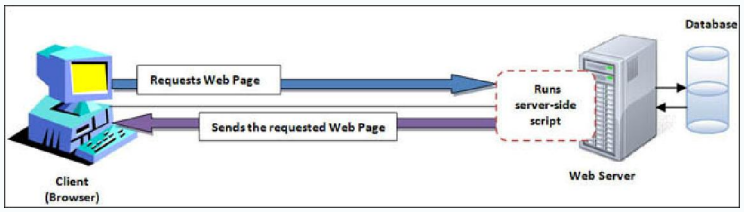
Server-side Scripting

Server-side scripting refers to the scripts that are executed by the Web server on the basis of the user's request. Some of the languages used for creating server-side scripts are Server-side JavaScript (SSJS), Perl, PHP, and Visual Basic Script (VBScript).

Server-side scripts are executed on the Web server. In this case, the information needs to be collected from the Web browser on the client side and is passed to a

program or script that is executed on the Web server. The script executing on the Web server performs certain tasks, such as establishing database connection and verifying data sent by the Web browser from the client computer.

Server-side scripting allows database interaction and can be used to process client-side queries and store client data in the database. This feature enables users to share and access information with other users of an application or a server. The communication between a client and a server in case of server-side scripting is shown in the following figure.



The Communication in Server-side Scripting

JavaScript and VBScript can be used for client-side scripting as well as server-side scripting.

Identifying the Benefits of JavaScript

JavaScript provides the following important benefits:

- **Handle events**: JavaScript can be used to execute functions whenever an event is triggered. For example, when a user rolls the mouse over any image, its background changes.
- **Gather browser information**: JavaScript can be used to gather browser information, such as the browser name and version. This information can be useful for the server to respond to client requests.
- **Manipulate cookies**: JavaScript can be used to access and store user information, such as client preferences and authentication information of a client computer, in the form of cookies.

A cookie is a piece of data that is used to identify a user. It is stored in a user's Web browser and is sent from a Web server while the user is browsing a website.

Implementing JavaScript in Web Pages

Consider the scenario of the BookYourHotel website. While booking a hotel, the customers are requested to specify their area of interest by clicking a radio button against certain options, such as adventure sports, movies, spa, and site seeing. The customers can avail the selected facility free of cost. If the customer selects any of these radio buttons, the details about that facility should open as a list. For example, if a customer selects the adventure sports radio button, the type of sports, such as paragliding, bungee-jumping, or river rafting, should be displayed as a list of check boxes, where the customer can select the preferred activity. Such functionalities can be implemented by using JavaScript.

JavaScript is one of the most popular scripting languages. It can be used to provide functionality to a Web page, such as populating a text box when a user selects an option in a list box. These functionalities can be triggered by the user action.

A script can be embedded directly into a Web page by writing the JavaScript code inside the `<SCRIPT>` tag or by writing the entire JavaScript code in an external JavaScript (.js) file. While using an external file for the JavaScript code, you need to refer to this file on the Web page.

Embedding a Script into a Web Page

The JavaScript code can be embedded into a Web page by using the `<SCRIPT>` tag. The following syntax is used for embedding a script into a Web page:

```
<SCRIPT type="text/javascript">
JavaScript statements
</SCRIPT>
```

In the preceding syntax, the `<SCRIPT>` and `</SCRIPT>` tags indicate the start and end of the script, respectively. The `type` attribute of the `<SCRIPT>` tag indicates the type of scripting language. One or more JavaScript statements between the `<SCRIPT>` tag and the `</SCRIPT>` tag form a script block. The script block is executed by the browser's built-in JavaScript interpreter.

The `<SCRIPT>` tag can be inserted into the body section of a Web page or the head section of a Web page or both. If the script is meant to be executed in response to an action performed by the user, it is normally placed in the head section. It helps in placing all scripts at one place without interfering with the content of the page. However, if the script needs to be executed as soon as the page is loaded, it is placed in the body section of the Web page.

Consider the following code:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
alert('Welcome to JavaScript');
</SCRIPT>
</BODY>
</HTML>
```

The preceding code illustrates the use of the `<SCRIPT>` tag in the body section of a Web page. It will display the message, **Welcome to JavaScript**, in a message box as soon as the Web page is loaded.

Creating and Using an External File

When you embed a script in an HTML page, it becomes difficult to manage both, the HTML code and the JavaScript code. To avoid this issue, you can write the JavaScript code in an external file and refer to the same in an HTML file. The external JavaScript file is saved with the .js extension. The following syntax is used to refer to an external JavaScript file:

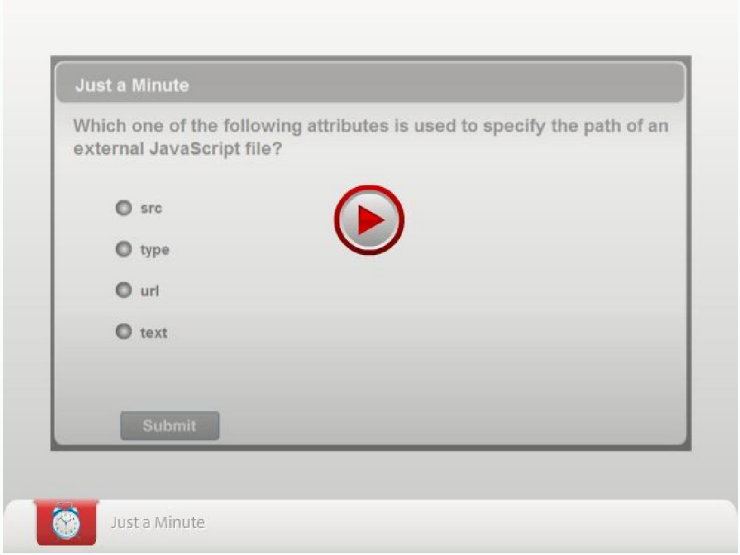
```
<SCRIPT type="text/javascript"
src="URL">
```

In the preceding syntax, URL refers to the path of the external JavaScript file.

Consider the example of the ShopForYou.com website. When a user visits the home page of the website, the list of products available for sale needs to be displayed in a message box. You can write the code to display the list of products in an external JavaScript file and add a reference to the file in the home page of the website.

To create and use the external JavaScript file, you need to perform the following steps:

1. Open Notepad and write the following code:
   ```
   alert( " PRODUCTS ON SALE : \n"
   +" 1. LEO Mobile \n" +" 2. LEO
   Camera\n" + " 3. RED shoes \n"+"
   4. KP Watch \n");
   ```
2. Save the file as **sale.js**.
3. Create the home page of the website in which you want to refer to the external JavaScript file. For this, write the following code in a Notepad file:
   ```
   <!DOCTYPE HTML>
   <HTML>
   <BODY>
   <H1>Buy Products</H1>
   </BODY>
   </HTML>
   ```
4. Save the file as **home.html**.
5. Add the highlighted code snippet, as shown in the following code in the **home.html** file:
   ```
   <!DOCTYPE HTML>
   <HTML>
   <BODY>
   <H1>Buy Products </H1>
   <SCRIPT type="text/javascript"
   src="sale.js">
   </SCRIPT>
   </BODY>
   </HTML>
   ```
   The `src` attribute of the `<SCRIPT>` tag is used to specify the path to the external JavaScript file, **sale.js.** The path to be specified in the `src` attribute can be either absolute or relative.



The absolute path is the complete address of a file. For example, if the file, sale.js, is stored in the D: drive, its absolute path is D:\sale.js. The relative path is the path of the file with respect to the current working directory. For example, if the HTML file containing the reference to the external JavaScript file is stored in the same directory as the sale.js file, the relative path of the file is given as sale.js.

Identifying Rules and Conventions Used in JavaScript

Every programming/scripting language has its own set of rules and conventions. Some of the rules and conventions for JavaScript are:

- **Semicolons**: JavaScript does not compulsorily require a semicolon to indicate the end of a statement. However, it is a good practice to insert a semicolon after a scripting statement as it enhances the readability of the code. For example, while writing two statements in the same line, you can use a semicolon to separate the statements. The following code snippet illustrates the usage of semicolons:
- **Quotes**: JavaScript allows you to use either double-quotation marks (" ") or single-quotation marks (`' '`) to enclose a string of characters. The following code snippet illustrates the usage of double quotes:
  ```
  alert(" PRODUCTS ON SALE : \n"
  +" 1.LEO Mobile \n" +" 2. LEO
  Camera\n" + "3. RED shoes \n"+"
  4.KP Watch \n");
  ```
  The following code snippet illustrates the usage of single quotes:
  ```
  alert(' PRODUCTS ON SALE : \n'
  +' 1.LEO Mobile \n' +' 2. LEO
  Camera\n' + ' 3. RED shoes \n'+'
  4.KP Watch \n');
  ```
- **Case sensitivity**: JavaScript is a case-sensitive scripting language. This means that the

statement, a=b, and the statement, A=B, are treated differently by JavaScript.

- **Comments**: Comments are statements that are not executed by the interpreter but are used to enhance the readability and understandability of the code. JavaScript allows usage of single-line as well as multi-line comments. Single-line comments are indicated by using `//`and multi-line comments are indicated by using the symbols, `/*` and`*/.` The following code snippet illustrates the usage of a single-line comment:

Activity 4.1: Understanding Scripting

Using Variables, Operators, and Control Structures

Consider the scenario wherein you need to create a Web page for ShopForYou.com that allows users to specify whether or not to display the list of products in the ascending or descending order of their prices.

To implement the preceding functionality on a Web page, you need to compare the prices of these items and display them either in the ascending or descending order, as requested by the user. To perform such tasks, you need to write scripts that use variables, operators, conditional constructs, and looping constructs provided in JavaScript.

Defining Variables

Consider a situation where you have created a Web page that accepts the quantity and unit price of a product that users wish to purchase and displays the total price of the product. While reading the quantity provided by the user, you need to store this value so that it can be used to calculate the total price. In addition, you need to store the price of the product.

A variable is a named location in memory that is used to store a value. In the preceding example, if the user provides 5 as the quantity of a product that needs to be purchased and the price of the product is $250, you need two variables, one for storing the quantity and the other for the price. The following code snippet shows the assignment of values to variables:

```
quantity = 5
price = 250
```

Declaring a Variable

Before using a variable in your program, you should declare it first. JavaScript allows you to declare a variable by using the `var` keyword. The syntax to declare a variable is:

```
var var_name;
```

In the preceding syntax, `var_name` refers to the name of the variable and the variable is declared by using the `var` keyword.

The following code snippet declares a variable named `employeeName`:

```
var employeeName;
```

JavaScript, which is a loosely-typed language, allows you to initialize a variable without specifying its data type. Numeric, String, and Boolean are the commonly-used data types in JavaScript.

Variables are divided into two categories, local and global. Local variables are declared in a block of code, such as within the body of a function or looping constructs. These variables can be accessed only in the specific block. Global variables are declared outside any block of code, but within the script. These can be accessed anywhere within the script.

Assigning a Value to a Variable

Values can be assigned to a variable in the following ways:

- Assigning a value to a variable after its declaration
- Initializing a variable while declaring it
- Initializing a variable without declaring it explicitly

Assigning a Value to a Variable After Its Declaration

Consider the following code snippet to initialize a variable after its declaration:

```
var employeeName;
employeeName="Peter";
```

The preceding code snippet declares the variable, `employeeName`, and then assigns the value, `Peter`, to it.

Initializing a Variable While Declaring It

Consider the following code snippet to initialize a variable while declaring it:

```
var employeeName="Peter";
```

The preceding code snippet declares the variable, `employeeName`, and assigns it the value, `Peter`.

Initializing a Variable Without Declaring It Explicitly

JavaScript provides you the flexibility to use a variable without declaring it. When a variable is used in a script without declaring it explicitly, it is automatically declared when used for the first time. However, it is a good practice to declare a variable before using it in a script as it increases the readability of the program.

Consider the following code snippet to initialize a variable without declaring it:

```
employeeName="Peter";
```

In the preceding code snippet, the variable, `employeeName`, is assigned a value but not declared explicitly. Therefore, the statement, `employeeName="Peter"`, declares the variable,

employeeName, implicitly and assigns the value, Peter, to it.

An array is a special type of variable that is used to store multiple values arranged in a definite sequence. These values are stored in indexed locations within the array. Before using arrays, you need to declare them. The following syntax is used for declaring an array:

```
var category = new Array(3)
```

In the preceding syntax, an array named category is created with a size of three.

Consider the following code snippet to assign a value at each indexed location:

```
category[0] = "Soaps"
category[1] = "Oils"
category[2] = "Food Products"
```

The values stored in the array can be accessed by using the index location where they are stored. The following syntax is used for accessing the first element in the array:

```
category[0];
```

Using Operators

An operator is a set of one or more characters that is used for computations or comparisons. Operators can be used to modify the values stored in variables. The values or variables on which the operator acts are known as operands. An operand can be a literal or a variable. A literal is a constant value of any data type, such as a number, string, or boolean.

The following code illustrates the use of operators in a script:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
var employeeName;
var basicSalary;
var othersAllowance; var TotalSalary;
employeeName="Peter";
basicSalary=20000;
othersAllowance=1000;
TotalSalary=basicSalary
+othersAllowance;
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the operands used are variables employeeName, basicSalary, TotalSalary, and othersAllowance including string literal, Peter, and numeric literals, 20000 and 1000. The + operator performs the addition operation on the value contained in basicSalary and othersAllowance and the value is assigned to the variable, TotalSalary.

Identifying Operator Categories

You can use the following categories of operators in JavaScript:

- Arithmetic Operators
- Assignment Operator
- Arithmetic Assignment Operators
- Comparison Operators
- Logical Operators

Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on variables and literals. The following table describes the commonly-used arithmetic operators.

Operator	***Description***	***Example***
+	*Used to add two numbers.*	*X=Y+Z;* *If Y is equal to 20 and Z is equal to 2, X will have the value, 22.*
-	*Used to subtract two numbers.*	*X=Y-Z;* *If Y is equal to 20 and Z is equal to 2, X will have the value, 18.*
*	*Used to multiply two numbers.*	*X=Y*Z;* *If Y is equal to 20 and Z is equal to 2, X will have the value, 40.*
/	*Used to divide one number by another. Returns the quotient of the division.*	*X=Y/Z;* If Y is equal to 21 and Z is equal to 2, X will have the value, 10.5.
%	*Used to divide two numbers and return the remainder. The operator is called as modulus operator.*	*X=Y%Z;* *If Y is equal to 21 and Z is equal to 2, X will contain the value, 1.*

The Arithmetic Operators

Assignment Operator

An assignment operator (=) is used to assign a value or a result of an expression to a variable. For example, the expression, x=5, stores the value, 5, in the variable, x.

Arithmetic Assignment Operators

Arithmetic assignment operators are used to perform arithmetic operations and assign the value to the variable at the left side of the operator.

The following table describes the commonly-used arithmetic assignment operators and their usage.

Operator	***Usage***	***Description***
+=	*X+=Y;*	*Same as:* *X = X + Y;*
-=	*X-=Y;*	*Same as:* *X = X - Y;*

*=	X*=Y;	Same as: X = X * Y;
/=	X/=Y;	Same as: X = X / Y;
%=	X%=Y;	Same as: X = X % Y;

The Arithmetic Assignment Operators

Comparison Operators

Comparison operators are used to compare two values and perform an action on the basis of the comparison. Whenever you use a comparison operator, the resulting expression holds a boolean value.

The following table describes the usage of comparison operators.

Operator	Usage	Description	Example (Assumption: Value of X is 20 and the value of Y is 25)
<	expression1 < expression2	Used to check whether expression1 is less than expression2.	var Result; Result = X <Y; Result will have the value true
>	expression1 > expression2	Used to check whether expression1 is greater than expression2.	var Result; Result = X >Y; Result will have the value false
<=	expression1 <= expression2	Used to check whether expression1 is less than or equal to expression2.	var Result; Result = X <=Y; Result will have the value true
>=	expression1 >= expression2	Used to check whether expression1 is greater than or equal to expression2.	var Result; Result = X >=Y; Result will have the value false
==	expression1 == expression2	Used to check whether expression1 is equal to expression2.	var Result; Result = X ==Y; Result will have the value false
!=	expression1 != expression2	Used to check whether expression1 is not equal to expression2.	var Result; Result = X !=Y; Result will have the value true
===	expression1== =expression2	Used to check whether expression1 is equal to expression2 and is of the same type.	var Result; Result = X ===Y; Result will have the value false

The Comparison Operators

Logical Operators

Logical operators are used to evaluate complex expressions in which there is a need to evaluate a single expression or multiple expressions to assess the result. They return a boolean value. The following table describes the usage of logical operators.

Operator	Usage	Description	Example (Assumption: Value of Xis 20 and the value of Y is 25)
&&	expression1 && expression2	Returns true if both expression1 and expression2 are true.	(X >10 && Y> 20) Returns true
!	! expression	Returns true if the expression is false.	!(X == Y) Returns true
||	expression1 || expression2	Returns true if either expression1 or expression2 or both of them are true.	(X ==5 || Y==5) Returns false

The Logical Operators

Using Conditional Constructs

Consider a situation where you need to execute certain statement(s) in a script on the basis of some condition. For example, you want to check whether a given number is even or odd and display a message accordingly. For this, you need to make decisions in the script code and execute a different set of statements depending upon the decision taken.

You can do this by using conditional constructs. These constructs allow you to execute a selective statement or a block of statements based on the result of the expression being evaluated. The two conditional constructs in JavaScript are:

- The `if...else` construct
- The `switch...case` construct

The if...else Construct

The `if` statement in the`if...else` conditional construct is followed by a logical expression in parenthesis. This condition is evaluated and a decision is made on the basis of the result. The following statements depict the syntax of the `if...else` construct:

```
if (exp)
{
// Statements;
}
else
{
// Statements;
}
```

In the preceding syntax, the expression,`exp`, is evaluated. If the result is true, the statements inside the `if` construct are executed. If the result is false, the statements inside the `else` construct are executed.

Consider the example of a game where you need to validate the age of a player. If the age is greater than 12, the player is allowed to play the game. Otherwise, an appropriate message needs to be displayed. The following code snippet shows the usage of the `if...else` construct:

```
var age=5;
if (age<12)
{
alert('Sorry! This game is for
children above 12 Years');
}
else
{
alert('Play the Game');
}
```

The preceding code snippet checks whether the age of the player is less than 12. The condition in the if statement checks the age of the player. If the condition evaluates to true, the message, **Sorry! This game is for children above 12 Years**, is displayed. If the condition evaluates to false, the message, **Play the Game**, is displayed.

The switch...case Construct

Another conditional construct available in JavaScript is the `switch...case` construct. It is used when you need to evaluate a variable for multiple values.

The following code depicts the syntax for the `switch...case` construct:

```
switch(VariableName)
{
case ConstantExpression_1:
// statements;
break;
case ConstantExpression_2:
// statements;
break;
case ConstantExpression_n:
// statements;
break;
default:
// statements;
break;
}
```

When the `switch` statement is executed, the variable passed as a parameter to the `switch` statement is evaluated and individually compared with each constant expression specified with each case statement. If one of the constant expressions is equal to the value of the variable given in the `switch` statement, the control is passed to the statement following the matched case statement. A `break` statement is used to exit the `switch` statement. This prevents execution of the remaining case structures by ending the execution of the `switch...case` construct. If none of the cases match, the statements under the default statement are executed.

Consider the following code where you want to display the name of the day of the week depending on the value of a variable:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
var day="5";
switch(day)
{
case "1":
alert("Day is Monday");
break;
case "2":alert("Day is Tuesday");
break;
case "3":
alert("Day is Wednesday");
break;
case "4":
alert("Day is Thursday");
break;
case "5":
alert("Day is Friday");
break;
case "6":
alert("Day is Saturday");
break;
case "7":
alert("Day is Sunday");
break;
default:
alert("Not a valid number");
break;
}
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the `switch` statement is used to evaluate and compare the value of the variable, `day`, with the case constants and display the name of the day.

Using Loop Constructs

Loop structures are used to repeatedly execute one or more lines of code. In JavaScript, the following loop structures can be used:

- The `while` loop
- The `do...while` loop
- The `for` loop

The while Loop

The `while` loop is used to repeatedly execute a block of statements till a condition evaluates to true. The `while` statement always checks the condition before executing the statements in the loop. The syntax for the `while` loop construct is:

```
while (expression)
{
statements;
}
```

In the preceding syntax, the expression is evaluated. If the result is true, the statements in the body of the loop are executed. Once all the statements in the block are executed, the control passes back to the loop and the expression is re-evaluated. The loop exits when the expression evaluates to false.

The following code illustrates the use of the `while` loop:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
var num=0;
while(num<20)
{ num=num+1; alert(num);
}
</SCRIPT>
</BODY>
</HTML>
```

The preceding code defines the variable, `num`, and initializes the same to the value, `0`. The `while` statement checks whether the value of `num` is less than

20. If the condition evaluates to true, the statements within thewhile loop are executed. This process continues till the value of var is less than 20.

The do...while Loop

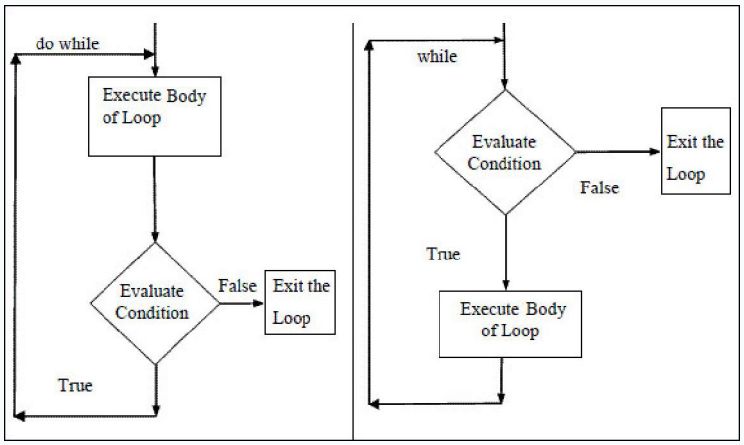
The do...while loop construct is similar to the while loop construct. However, the statements within the do...while loop are executed at least once, in comparison to the while loop, where the statements in the block are not executed when the condition evaluates to false. In addition, the statements within the do-while loop are executed before the condition is checked as compared to the while loop, where the statements within the block are executed after the condition is checked. The following syntax is used to declare the do-while loop:

```
do
{ Statements;
}
while(condition)
```

Consider the following code that illustrates the use of the do-while loop:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
var num=0;
do
{
num=num+1;
alert(num);
}
while(num<20)
</SCRIPT>
</BODY>
</HTML>
```

The difference between the functionality of the do...while loop construct and that of the while loop construct is shown in the following figure.



The Functioning of the do...while and while Loops

The for Loop

The for loop allows the execution of a block of code depending on the result of the evaluation of the test condition. The following syntax is used to declare the for loop:

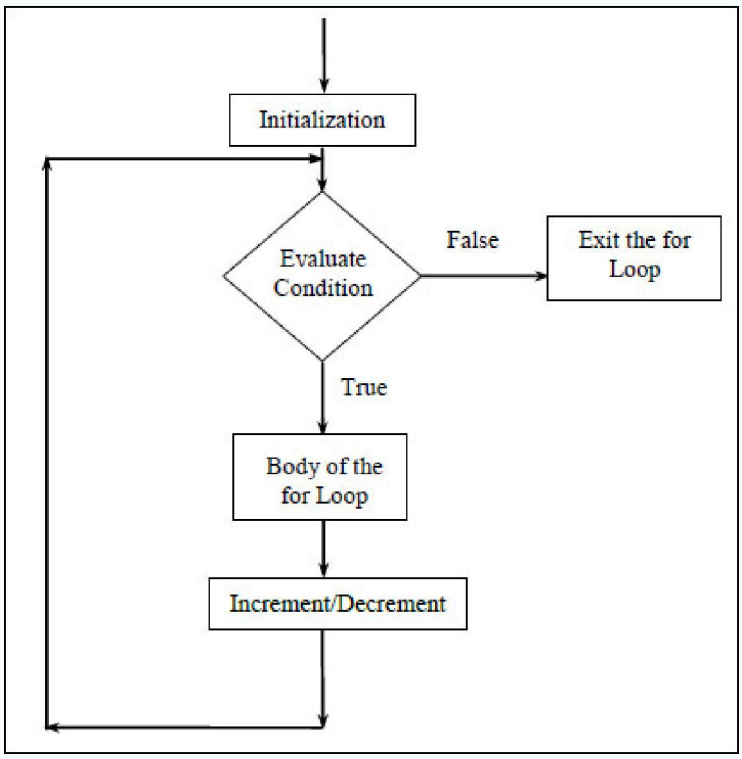
```
for (initialize variable; test
condition; step value)
{
// code block
}
```

In the preceding syntax:

- q initialize variable: Initializes the loop variable to a value.
- q test condition: Specifies the condition to be checked before executing statements in the code block.
- q step value: Indicates the increment or decrement to be performed for every iteration.

The execution of the for loop starts with the initialization of the loop variable. It, then, evaluates the test condition. If the test condition evaluates to true, the code block in the body of the loop is executed. If it evaluates to false, the for loop is exited. Once all the statements in the code block are executed, the variable is incremented or decremented as specified in the step value. Once the value is iterated, the test condition is re-evaluated. This process continues till the test condition evaluates to true.

The sequence of execution of the for loop is shown in the following figure.



The Sequence of Execution of the for Loop

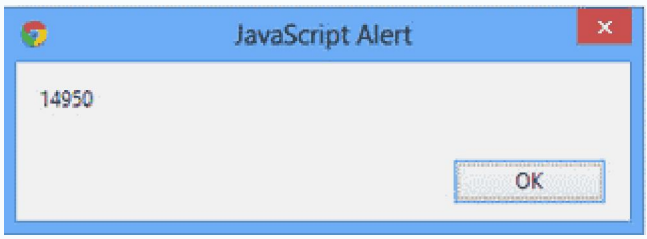
The following code illustrates the use of the for loop:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
var num;
var sum=0;
for(num=100;num<200;num=num+1)
{
```

```
sum=sum+num;
}
alert(sum);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code creates variables named `num` and `sum`. In the `for` loop, the value, `100`, is assigned to the `num` variable. The condition inside the `for` loop checks whether the value of `num` is less than `200`. If the condition evaluates to true, the statements within the `for` loop are executed. This process continues till the value of the `num` variable is less than `200`.

The output of the preceding code will be displayed, as shown in the following figure.



The Code Output

The working of the `while` loop is similar to that of the `for` loop. The main difference is that the variable is incremented within the code block of the `while` loop as compared to the `for` loop, where the variable is incremented after all the statements in the code block are executed.

Break and Continue Statements

In some situations, there may be a need to exit a loop before the loop condition is checked after iteration. The `break` statement is used to exit the loop. It prevents the execution of the remaining statements of the loop. The `break` statement is usually placed within an `if` construct inside the loop. The `continue` statement is used to skip all the subsequent instructions and take the control back to the beginning of the loop.

The following syntax is used to show the use of the `break` statement inside the `while` loop:

```
while(test condition)
{
//Statement1;
if (test condition1)
{
break;
}
//Statement2;
//Statement3;
}
```

In the preceding syntax, if `test condition` is true, the control enters the body of the loop. It executes `statement1`. Next, it checks test condition1. If it evaluates to true, the control is transferred outside the loop. As a result, `Statement2` and `Statement3` are not executed. If `test condition1` evaluates to false, the loop continues its normal course of execution. As a result, `Statement2` and `Statement3` are executed.

Consider the following code that illustrates the use of the `break` and `continue` statements:

```
<!DOCTYPE HTML>
<HTML>
<HEAD></HEAD>
<BODY>
<SCRIPT type="text/javascript">
var iNum = 0;
for (var i=1; i < 10; i++)
{
if (i % 3 == 0)
{
break;
}
iNum++;
}
alert(iNum);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code creates the variable, `iNum`, and assigns the value, `0`, to it. The `for` loop iterates the variable, `i`, from 1 to 10. The `if` statement in the `for` loop validates if the value of `i` is divisible by `3`. If `i` is divisible by `3`, the `break` statement is executed and the control comes out of the `for` loop. When the preceding code is executed, the value, `2`, is displayed.

However, if the preceding code is executed by replacing `break` with the `continue` statement, a different output is displayed. For example, consider the following code:

```
<!DOCTYPE HTML>
<HTML>
<HEAD></HEAD>
<BODY>
<SCRIPT type="text/javascript">
var iNum = 0;
for (var i=1; i < 10; i++) {
if (i % 3 == 0) {
continue;
}
iNum++;
}
alert(iNum);
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the `if` statement in the `for` loop validates if the value of `i` is divisible by `3`. If `i` is divisible by `3`, the `continue` statement is executed and the control comes back to the beginning of the `for` loop. When the preceding code is executed, the value,`6`, is displayed.

Implementing Functions

Consider the ShopForYou.com website. The company is required to calculate the tax levied on the products purchased by the customers. The value of tax varies from item to item. For example, the tax on certain products, such as medicine, is 0%, and the tax on mobile products is 12.5%. The code that calculates the tax on various categories of products needs to be used at various locations in the script of your page. If the code is repeated at all these locations, any change required in the code will need to be made at multiple locations. For example, if you want to change the tax rate from 12.5% to 10%, you will need to make this change at all the locations where this code is used. In addition, if any reused piece of code contains an error, the error needs to be corrected at multiple locations.

To overcome this problem, functions are introduced. You can write the code that needs to be reused inside a function. Now, instead of repeating the use of code, you can just make a call to the function that contains the required code. This enables you to write the code only once and use it whenever required. Thus, functions are used to optimize the performance by implementing the concept of reusability.

Introducing Functions

A function is a self-contained block of statements that has a name. Functions can be executed whenever the same code is required to be performed repeatedly at different locations in a script.

A function is created as a separate module with a name attached to it. Each function can have several statements within it. When functions are used, you can easily detect the error in the script by comparing the expected result of execution of each function with the actual result of execution of that function. If the results differ, it means that there is an error in the code. Thus, it becomes easier to debug the code.

In other words, functions make your code modular, simple, and reusable. In JavaScript, functions are of the following types:

- Built-in functions
- User-defined functions

Built-in Functions

Built-in functions are ready to use as they are already coded. Some of the built-in functions supported by JavaScript are:

- `isNaN()`
- `parseInt()`
- `parseFloat()`
- `eval()`
- `prompt()`
- `confirm()`

`isNaN()`

The `isNaN()` function determines whether a parameter is not a number. The function returns true if the parameter is not a number.

The following code illustrates the use of the `isNaN()` function:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
num1="Two thousand"; num2=8000;
Result=isNaN(num1); alert(Result);
Res=isNaN(num2); alert(Res);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code creates the variable, `num1`, and assigns the value, `Two thousand`, to it. It also creates a variable, `num2`, and assigns the value, `8000`, to it. The `isNaN()` function evaluates whether the variables, `num1` and `num2`, contain numeric data or not. As the value in the `num1` variable is a string, the `isNaN(num1)` function will return true. However, the value in `num2` is numeric. Therefore, the `isNaN(num2)` function will return `false`.

NaN stands for Not a Number.

`parseInt()`

The `parseInt()` function parses the string parameter and returns the corresponding integer.

Consider the following code snippet:

```
x="5";
y=parseInt(x);
```

In the preceding code snippet, the variable, `x`, stores the string, `5`. The `parseInt()` function takes the parameter, `x`, parses it and assigns the integer value, `5`, to the variable, `y`.

`parseFloat()`

The `parseFloat()` function takes a string parameter and returns a floating point number. Consider the following code snippet:

```
x="6.2";
y=parseFloat(x);
```

In the preceding code snippet, the variable, `x`, stores the string, `6.2`. The `parseFloat()` function takes the parameter, `x`, parses it, and assigns the floating point number, `6.2`, to the variable, `y`.

`eval()`

The `eval()` function is used to evaluate or execute a parameter. If the parameter is an expression, the expression is evaluated. However, if the parameter is a JavaScript statement, the statement is executed.

Consider the following code:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
```

```
var res1=0; res1=eval(5+10); alert
(res1);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code evaluates the expression, `5+10`, and displays the result.

Consider the following code:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
eval("num1=5;num2=10;res1=num1
+num2;alert(res1);");
</SCRIPT>
</BODY>
</HTML>
```

The preceding code evaluates the JavaScript statements written inside the `eval()` function and displays the result in the message box.

`prompt()`

The `prompt()`function is used to display a prompt dialog box, which allows a user to input a value. The prompt dialog box contains two buttons,**OK** and **Cancel**. If the user clicks the **OK** button, the `prompt()` function returns the value entered by the user. However, if the user clicks the **Cancel** button, a null value is returned.

The following code illustrates the use of the `prompt()` function:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
var name=prompt("Please Enter Your
Name","John");
alert(name);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code prompts a user to enter the name by using the `prompt()` function and displays the name in a message box. The `prompt()` function contains two parameters. The first parameter, **Please Enter Your Name**, asks the user to provide the name and the second parameter specifies **John** as the default name. The first parameter displays a message in a dialog box, and it is a required parameter. On the other hand, the second parameter specifies the default text and it is an optional parameter.

The `prompt()` function always returns a string value. Therefore, if you need to work with data other than string, you need to cast or convert the string value that was returned by the `prompt()` function.

The following code shows how to convert the string value returned by `prompt()`:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT>
var a,b,c;
a=parseInt(prompt("Enter number
1:"));
b=parseInt(prompt("Enter number
2:"));
c=a+b;
document.write(c);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code prompts the user to enter two numbers, and then converts them by using the `parseInt()` function.

The `document.write()` function is used in JavaScript to display a string output directly on the Web page. You will learn about it in detail in the coming chapters.

`confirm()`

The `confirm()` function is used to display a dialog box that will enable a user to verify or accept a task. This dialog box contains two buttons, **OK** and **Cancel**. If a user clicks the **OK** button, the `confirm()` function returns true. If the user clicks the **Cancel** button, the `confirm()` function returns false.

The following code illustrates the use of the `confirm()` function:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
var response;
response=confirm("Do You Wish to
Continue");
if(response==true){
alert("You can proceed further");
}
else
{
alert("You cannot proceed further");
}
</SCRIPT>
</BODY>
</HTML>
```

If the user clicks the **OK** button, the preceding code displays the message, **You can proceed further**. However, if the user clicks the **Cancel** button, the message, **You cannot proceed further**, is displayed.

User-defined Functions

JavaScript enables you to define your own functions according to your needs. For example, you need to calculate the average marks of students by accepting the marks from them. In this case, you can create a user-defined function, `Average()`, and call this function whenever you need to calculate the average marks. It

enhances the modularity and efficiency of the code.

Creating Functions

Functions are created by using the keyword, `function`, followed by the function name and the parentheses, `()`. A function is normally defined in the head section of a Web page.

The following syntax is used to create functions:

```
function [functionName] (Variable1,
Variable2)
{
//function statements
}
```

A user-defined function can optionally accept a list of parameters. The parameters that a function accepts are provided in parentheses and separated by commas. In the given code, `Variable1` and `Variable2` represent the parameters passed to the function. The function can use the values passed in these parameters to perform certain operations.

Consider the following code snippet to illustrate the creation of a function:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
function tax
(product_category,product_name,price)
{
if(product_category=="Mobile")
{
total_price=(12.5/100)*price+price;
alert("Total price of " +product_name
+" is: $" +total_price);
}
if(product_category=="Medicine")
{
total_price=price;
alert("Total price of " +product_name
+" is: $" +total_price);
}
}
</SCRIPT>
</HEAD>
<BODY>
. . . .
</BODY>
</HTML>
```

In the preceding code snippet, the `tax()` function is created that accepts three parameters,`product_category`, `product_name`, and `price`. The `tax()` function is used to calculate the total price of a product after adding the tax amount to the price of the product.

Accessing Functions

Once you have created a function, you can call the same from any portion of the code where you want to use the functionality provided by the function. A function can return a value of any data type to the calling code. Let us see how a function is called, how a value from a function is returned, and how the value returned by a function is retrieved.

Calling a Function

A function is called by using the function name. The following syntax is used for accessing a function:

```
functionName ();
```

In the preceding syntax, `functionName` is the name of the function.

While calling a function, you can also pass a variable or a value to it. The variable or the value passed to a function is known as a parameter. If parameters need to be provided, these are specified in parenthesis while calling the function. The following syntax is used for specifying parameters in a function:

```
functionName (parameter1, parameter
2...);
```

The following code illustrates how a function is accessed:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
. . .
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT type="text/javascript">
product_category="Medicine";
tax
(product_category,"Paracetamol",8000)
; product_category="Mobile";
tax(product_category,"GV3",3000); tax
(product_category,"XV5",5000);
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the `tax()` function is called and `product_category`, `product_name`, and `price` are passed as parameters. The `tax()` function displays the total price after adding the tax amount to the price of the product. If `product_category` is `Medicine`, no tax amount is added to the price of the product. However, if `product_category` is `Mobile`, 12.5 % of the product price is added as the tax.

Returning Values from a Function

A function can return a value through the `return` statement. The following code illustrates how a value from a function is returned:

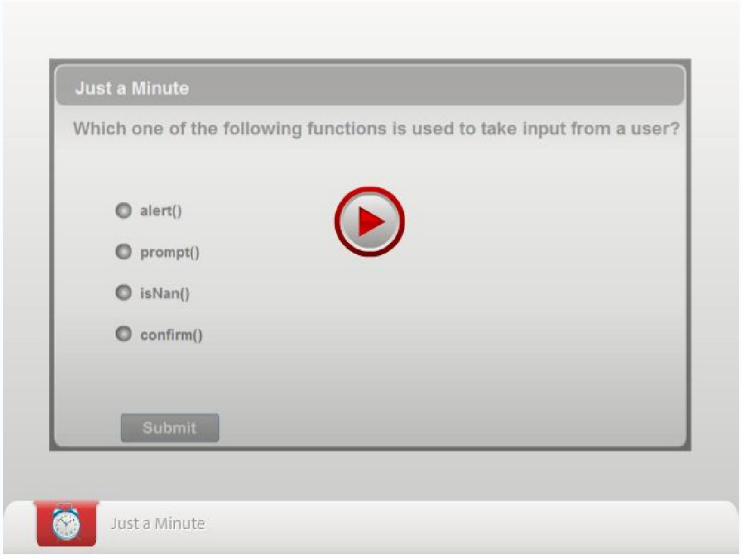
```
function functionName()
{
var variable=10;
return variable;
}
```

Consider the following code:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
function sum(a,b)
{
return a+b;
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT type="text/javascript"> total
= sum (3, 6); document.write(total);
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the `sum()` function accepts two numbers, `3` and `6`, as parameters and returns their sum, 9.

A function can accept more than one parameter but can return only one value.



Activity 4.2: Implementing Functions

Summary

In this chapter, you learned that:

- To create a dynamic and interactive Web page, you need to incorporate a block of code, which is known as script, in the Web page.
- Client-side scripting refers to the scripts that are executed at the client-side by the Web browser, running on the user's computer.
- Server-side scripting refers to the scripts that are executed by the Web server on the basis of the user's request.
- A script can be embedded directly into a Web page by writing the JavaScript code inside the `<SCRIPT>` tag or by writing the entire JavaScript code in an external JavaScript (.js) file.
- The external JavaScript file is saved with the .js extension.
- Values can be assigned to a variable in the following ways:
  - Assigning a value to a variable after its declaration
  - Initializing a variable while declaring it
  - Initializing a variable without declaring it explicitly
- An operator is a set of one or more characters that is used for computations or comparisons.
- You can use the following categories of operators in JavaScript:
  - Arithmetic operators
  - Assignment operators
  - Arithmetic Assignment Operators
  - Comparison operators
  - Logical operators
- The two conditional constructs in JavaScript are:
  - The `if...else` construct
  - The `switch...case` construct
- In JavaScript, the following loop structures can be used:
  - The `while` loop
  - The `do...while` loop
  - The `for` loop
- A function is a self-contained block of statements that has a name.
- Some of the built-in functions supported by JavaScript are:
  - `isNaN()`
  - `parseInt()`
  - `parseFloat()`
  - `eval()`
  - `prompt()`
  - `confirm()`
- Functions are created by using the keyword, `function`, followed by the function name and the parentheses, `()`.

Reference Reading

Understanding Scripting

Reference Reading: Books	*Reference Reading: URLs*
JavaScript: the complete reference By Thomas A. Powell, Fritz Schneider	*http://www.w3schools.com/web/web_javascript.asp* *http://www.w3schools.com/web/web_scripting.asp*

Implementing JavaScript in Web Pages

Reference Reading: Books	*Reference Reading: URLs*
The Definitive Guide to HTML5 By Adam Freeman	*http://www.w3schools.com/js/default.asp*

Using Variables, Operators, and Control Structures

Reference Reading: Books	*Reference Reading: URLs*
The Definitive Guide to HTML5 By Adam Freeman	*http://softearth.tripod.com/Books/Using_Javascript/ch2.htm*

Implementing Functions

Reference Reading: Books	*Reference Reading: URLs*
The Definitive Guide to HTML5 By Adam Freeman	*http://www.w3schools.com/js/js_functions.asp*

Chapter 5

Creating Dynamic Web Pages

Today, most of the Web pages, such as the registration and login pages, have some way of accepting input from a user. To design such Web pages, you need to use several elements, known as form elements, which enable the user to input values. The data entered using these form elements needs to be processed further upon submission. Every Web page should respond to end user actions, such as clicking a submit button, changing a value in a field, or making a selection from a list. All these actions are referred to as events and need to be handled appropriately. This dynamic functionality can be implemented using the JavaScript object model.

This chapter discusses how to design an HTML form for accepting the user input. In addition, it discusses the various types of browser and form objects.

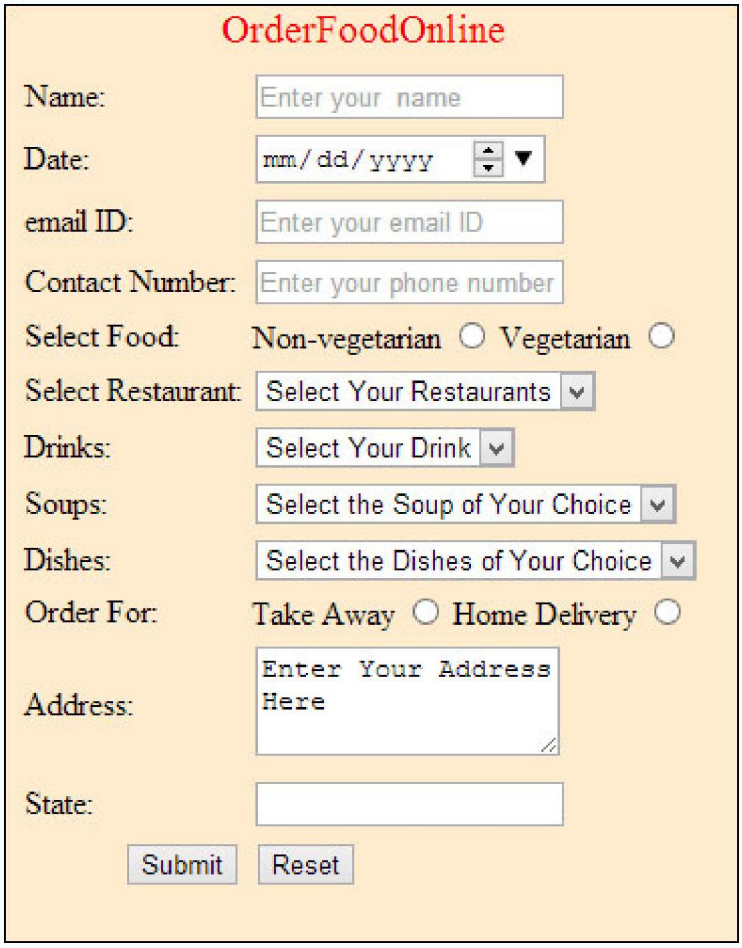
Objectives

In this chapter, you will learn to:

- Design an HTML form
- Manipulate the components of a Web page

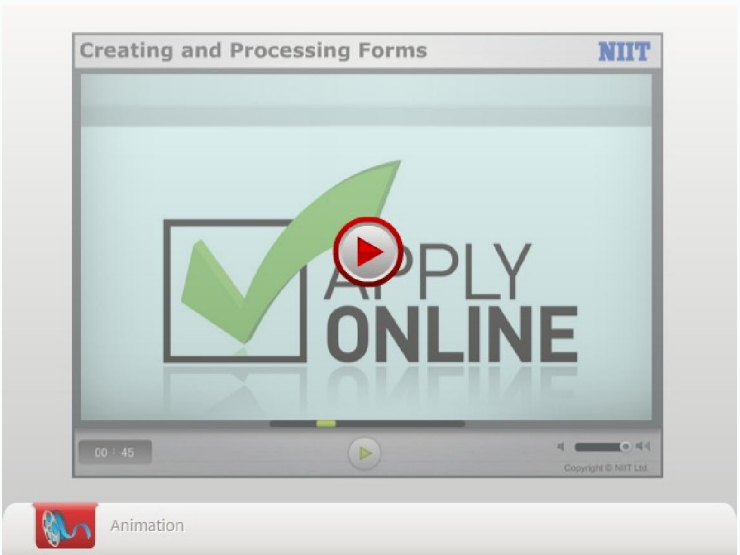
Designing an HTML Form

Consider the scenario of FoodExpress.com. This website facilitates customers to place online orders with various restaurants. For this, the customers need to provide various details, such as the name, contact number, and address, in an online form named as **OrderFoodOnline**, as shown in the following figure.



The OrderFoodOnline Form

Forms are interactive Web pages that are used to accept the user input. These forms consist of one or more types of input fields, such as text fields, check boxes, radio buttons, and the submit buttons. These input fields enable the user to fill and submit the information to a website. HTML supports various tags to create forms and input fields.



Creating Forms

To create a form on a Web page, you need to use the `<FORM>` tag. The `<FORM>` tag helps you define a form. It has an opening `<FORM>` tag and a closing `</FORM>` tag. The following syntax is used to specify the `<FORM>`

tag:

```
<FORM [attribute list]>
..
..
</FORM>
```

The <FORM> tag supports the following attributes:

- name
- ID
- action
- method
- autocomplete
- novalidate
- target

name

The name attribute is used to specify a unique name for a form. It is used to uniquely identify a form in the get or post methods at the time of form submission. In addition, multiple forms can be present on a Web page. These forms can be differentiated using the name attribute. The following syntax is used to specify a name for a form:

```
<FORM name= "User defined name">...
</FORM>
```

ID

The ID attribute is used to specify a unique ID for the form element on a Web page. The ID attribute should be unique in the entire HTML document. Both the ID and the name attributes are used to uniquely identify a form element on a Web page. However, the ID given to a form element is used when you need to reference it with style sheets or scripts. You can assign a unique ID to a form element by using this attribute.

The following syntax is used to specify an ID for a form:

```
<FORM ID= "User defined ID">... </
FORM>
```

action

The action attribute specifies the URL of the page to which the contents of the form are submitted. If this attribute is missing, the URL of the document itself is assumed as the location for the form submission. The following syntax is used for the action attribute to specify the URL of the page:

```
<FORM action="filename or URL">
```

method

The method attribute is used to specify the format in which the data will be submitted to the file or the URL specified in the action attribute. It can take either of the following values:

- get
- post

The default value of the method attribute is get.

get

The get value appends the form data to the URL of the form as the name value pair at the time of form submission. Since the data is appended to the URL, it is always visible to the users. Therefore, it is not a correct approach when you need to submit the sensitive data. Also, the size of data that can be submitted using the get method is limited to only 3000 characters.

post

The post value does not append the form data to the URL of the form when it is submitted. Therefore, the data is not shown in the URL and offers a secure way of submitting the data. Also, a large amount of data can be sent using the post method. This method can be used to send textual as well as image data.

autocomplete

The autocomplete attribute is used to specify whether a form should have the autocomplete feature on or off. If it is on, the browser automatically completes the values in the fields based on the values that the user has entered before. The following code snippet is used to specify the autocomplete attribute for a form:

```
<FORM ID="fileID" autocomplete= "on">
```

novalidate

The novalidate attribute specifies that the data in the form should not be validated by the browser at the time of data submission. It is an empty attribute that does not contain any value. The following code snippet is used to specify the novalidate attribute for a form:

```
<FORM ID="fileID" novalidate>
```

target

The target attribute is used to specify the name of the frame or the window in which the response obtained after submitting the form needs to be displayed. The following syntax is used to specify thetarget attribute for a form:

```
<FORM target="_blank|_self|_parent|
_top|frame_name">
```

The target attribute can have any one of the following values:

- _blank: Specifies that the response should be displayed in a new frame or window.
- _self: Specifies that the response should be displayed in the same frame.
- _parent: Specifies that the response should be displayed in the parent frame or window.
- _top: Specifies that the response should be displayed in the full body of the window.
- frame_name: Specifies that the response should be displayed in the specified frame.

In the BookYourHotel scenario, the following code is used to create the **OrderFoodOnline** form:

```
<FORM name= "OrderFoodOnline" ID=
"Order_Food" method= "post">
......
</FORM>
```

Exploring Form Elements

You need to design the **OrderFoodOnline** form further so that it can be used to accept the user input. For this,

you need to add various input fields on the form. These fields can be added to a form by using the following tags:

- <INPUT>
- <SELECT>
- <LABEL>
- <FIELDSET>
- <TEXTAREA>
- <DATALIST>
- <KEYGEN>
- <OUTPUT>
- <BUTTON>

<INPUT>

The <INPUT> tag is used to create input fields inside a form. These fields are used to accept input from the users. Input fields are of various types, such as text box, radio button, or check box. The type of input field is determined by the value of its type attribute. The <INPUT> tag has some important attributes, such astype, value, name, ID, autocomplete, autofocus, form, required, pattern, and placeholder.

Defining the type Attribute

The type attribute of the <INPUT> tag defines the type of input field to be added on the form. The type attribute has the following values:

- text: Creates a single line editable text field. When the value of the type attribute is text, two additional attributes, size and maxlength, can also be specified. The size attribute is used for limiting the character width of the text field in the form. The maxlength attribute defines the maximum number of characters that can be typed in the text field. The following code snippet is used to create a text field:

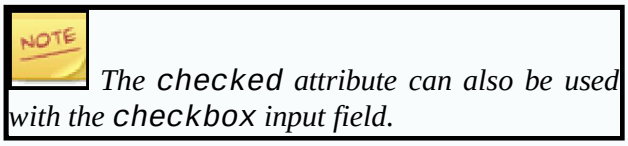
```
First Name: <INPUT type="text"
name="fname" size="20"
maxlength="20">
Last Name: <INPUT type="text"
name="lname" size="20"
maxlength="20">
```

  In the preceding code snippet, two text fields, fname and lname, are created. Both the fields are 20 characters wide and can accept a maximum of 20 characters.
- password: Creates a password field, which will not display the characters being typed by the user. It hides the actual characters and shows the masked values for each character, such as ****. The following code snippet is used to create a password field:

```
<INPUT type="password"
name="accountpasswrd">
```

- radio: Creates a radio button, which lets the user select one of the options from a set of given options. When the value of the type attribute is radio, an additional attribute, checked, can also be specified. The checked attribute is used to specify that the radio button appears pre-selected when the page loads. The following code snippet is used to create a radio field:

```
<INPUT type="radio" name="Rating
of Hotel" checked>
5-Star<BR/>
<INPUT type="radio" name="Rating
of Hotel"> Budgeted
```



The checked attribute can also be used with the checkbox input field.

In the preceding code snippet, two radio buttons 5-Star and Budgeted, are created. The checked attribute is applied on the first radio button, 5-Star. Therefore, it appears selected by default.

The radio buttons in a group are given the same name to ensure that the user is able to select only one radio button at a time.

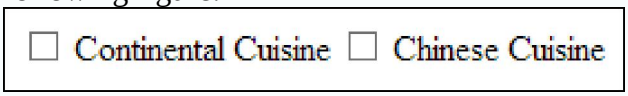
The radio buttons are displayed as shown in the following figure.

The Radio Buttons

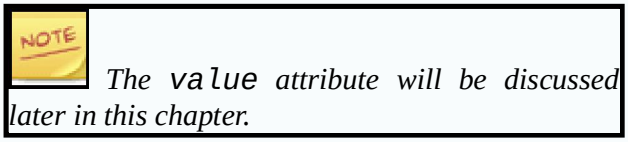
- checkbox: Creates a check box, which lets the user select one or more options from a set of given options. The following code snippet is used to create the check box fields:

```
<INPUT type="checkbox"
name="Cuisine1"
value="Continental Cuisine">
Continental Cuisine
<INPUT type="checkbox"
name="Cuisine2" value="Chinese
Cuisine"> Chinese Cuisine
```

  In the preceding code snippet, two check boxes, Continental Cuisine and Chinese Cuisine, are created, as shown in the following figure.



The Check Boxes



The value attribute will be discussed later in this chapter.

- submit: Creates a submit button, which submits the form data to the location specified in the action attribute of the form. When the value of the type attribute is submit, some additional attributes, such asformaction, formmethod, formtarget, and formnovalidate can also be specified. The

description of these attributes is given in the following list:

- `formaction`: It is used to specify a URL where the form data would be submitted when the submit button is clicked. The URL specified in the `formaction` attribute of the submit button overrides the URL specified in the `action` attribute of the `<FORM>` tag. Therefore, the form is forcibly submitted to the URL specified in the `formaction` attribute of the submit button, instead of the URL specified in the `action` attribute of the form.
- `formmethod`: It is used to specify the method, such as `get` and `post`, using which the form data will be sent to the file or URL specified in the `action` attribute of the form. The value specified for the `formmethod` attribute of the submit button overrides the value of the `method` attribute of the `<FORM>` tag.
- `formtarget`: It is used to specify the name of the frame or the window in which the response would be displayed when the form is submitted. The value specified for the `formtarget` attribute of the submit button overrides the value of the `target` attribute of the `<FORM>` tag.
- `formnovalidate`: Every form is validated by default, unless you use the `novalidate` attribute with the `<FORM>` tag. The `formnovalidate` attribute of the submit button is used to force the form to behave like a form with the `novalidate` attribute.

The `formaction`, `formmethod`, `formtarget`, and `formnovalidate` attributes can also be used with the `INPUT type`, `image` and the `<BUTTON>` tag.

The `INPUT type`, `image` and the `<BUTTON>` tag will be discussed later in this chapter.

- `reset`: Creates a reset button, which clears the values entered by a user in the form fields. The following syntax is used to create a reset field:

```
<INPUT type="reset"
name="reset">
```

- `URL`: Adds a field that is used to enter the URL of a website. The value in this field is automatically validated for correctness when the form is submitted. The following syntax is used to create a URL field:

```
<INPUT type="url"
name="locator">
```

URL stands for Uniform Resource Locator. It is the address of a unique file or resource available on the web that is accessible through the Internet. www.w3schools.com is an example of a URL specified in the address bar of a browser to connect to the home page of w3schools.com website.

- `email`: Creates a field in an HTML form to accept the email address from the users. You can even use a normal text field to accept the email address. But every email address needs to be validated for its correct format, <username>@<domainname>. If you use the text field to accept an email address, such a validation is difficult to perform. However, with an email field, the email address is automatically validated upon submitting the form.

  When the value of the `type` attribute is `email`, an additional attribute, `multiple`, can also be specified. The `multiple` attribute is used to specify that the user can enter more than one value in the field. The following code snippet is used to create an email field:

```
<INPUT type="email"
name="email_id" multiple>
```

- `range`: Creates a slider control to enter a numeric value within a range. The default range of the slider is 0 to 100. When the value of the `type` attribute is `range`, additional attributes, such as`min`, `max`, `value`, and `step`, can also be specified. The `min` attribute is used to specify the minimum value in a range. The `max` attribute is used to specify the maximum value in a range. The `value` attribute stores the current value associated with the range field. The `step` attribute is used to specify a number with which the `value` of the range field will increase or decrease as you move the slider. The following code snippet is used to create a range field:

```
<INPUT type="range" max="50"
min="10" step="5" value="10">
```

  The preceding code snippet creates a range with the default value `10`, the minimum value, `10`, and the maximum value, `50`. The value will increase by `5` within the minimum and maximum limit when you use the slider to specify the numeric value. The range field is displayed, as shown in the following figure.

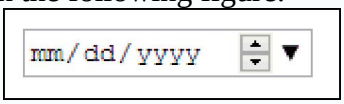
The Range Field

The `min`, `max`, *and* `step` *attributes can also be used with the input types, such as* `number`, `date`, *and* `time`. *The input types, such as* `number`, `date`, *and* `time`, *will be discussed later in this chapter.*

- `date`: Is used to define a date field in an HTML form. It allows a user to select a date. The following code snippet is used to create a date field:

```
<INPUT type="date" name="bday">
```

The preceding code snippet creates a date field, as shown in the following figure.

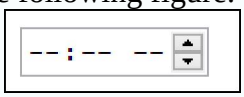


The Date Field

- `time`: Is used to define a time field in an HTML form. It allows a user to select time. The following code snippet is used to create a time field:

```
<INPUT type="time"
name="usr_time">
```

The preceding code snippet creates a time field, as shown in the following figure.



The Time Field

- `number`: Is used to create an input field for entering a numeric value. The following code snippet is used to create a number field:

```
<INPUT type="number"
name="quantity" min="0"
max="50">
```

The preceding code snippet creates a number field with a minimum value of 0 and a maximum value of 50. The number field is displayed, as shown in the following figure.

The Number Field

- `tel`: Is used to accept a telephone number from the user. The following code snippet is used to create a telephone field:

```
<INPUT type="tel" name="usrtel">
```

- `image`: Is used to specify an image to be used as a `submit` button. When the value of the `type` attribute is `image`, some additional attributes, such as `height`, `width`, `alt`, and `src` can also be specified. The following list describes these attributes:
  - `height` and `width`: The `height` and `width` attributes specify the height and width for the image.
  - `alt`: The `alt` attribute is used to specify text for the alternative buttons to be displayed when the image specified in the `src` attribute could not be used.
  - `src`: The `src` attribute specifies the URL of the image that will be used as a submit button.

The following code snippet is used to create an image field:

```
<INPUT type="image"
src="img_submit.gif"
alt="submit" width="48"
height="48" />
```

The preceding code displays the image to be used as the submit button, as shown in the following figure.

The Image Used as the Submit Button

Defining the value Attribute

The `value` attribute specifies the value of the field. It behaves differently for different input types, as shown in the following list:

- For `button`, `reset`, and `submit` input types, the `value` attribute defines the text that appears on the face of the buttons.
- For `text` and `password` input types, the `value` attribute defines the default value for the fields.
- For `checkbox`, `radio`, and `image` input types, the `value` attribute defines the values associated with the input fields. All the radio buttons in a group have a common name. Therefore, the `value` attribute is used to differentiate the radio buttons in a group. In addition when a form is submitted, the selected radio button is identified by using its `value` attribute. Similarly, the user can select one or more check boxes and submit the form. All the selected check boxes can be identified by using the `value` attribute.

The following code snippet is used to specify the `value` attribute:

```
<INPUT type="text" name="fname"
value="Enter Your First Name">
```

In the preceding code snippet, `Enter Your First Name`, is assigned to the `value` attribute of the input field.

Defining the name Attribute

The `name` attribute specifies the name for the input field. It is used to identify the form field when the form data is submitted. The following code snippet is used to specify a name for the input field:

```
<INPUT type="text" name="Text1" />
```

In the preceding code, the name, `Text1`, is assigned to the text field.

Defining the ID Attribute

The `ID` attribute provides a unique ID to the input field. This attribute is used to access the input fields in the CSS or JavaScript code.

The following code snippet is used to specify a unique ID to the input field:

```
<INPUT type="text" ID="value">
```

In the preceding code, the ID, `value`, is assigned to the input field.

Defining the autocomplete Attribute

The `autocomplete` attribute is used to specify whether a form element should have the autocomplete feature on or off. If it is on, the browser automatically completes the values in the form fields based on the values that the user has entered earlier. The following code snippet is used to specify the `autocomplete` attribute for a form:

```
<INPUT type="email" name="email"
autocomplete="on">
```

In the preceding code snippet, the `autocomplete` attribute is set to `on` for the `email` input field.

Defining the autofocus Attribute

The `autofocus` attribute is used to ensure that the form element has a focus when a Web page loads. The following code snippet is used to specify an `autofocus` attribute for an input field:

```
<INPUT type="text" name="lname"
autofocus>
```

In the preceding code snippet, the text field with the name, `lname`, will have the focus when the page loads.

Defining the required Attribute

The `required` attribute is used to specify that an input field must not be left empty while submitting the form. The `required` attribute can be used with the input types, such as `text`, `tel`, `email`, `password`, `number`, `checkbox`, and`radio`. The following code snippet is used to specify the `required` attribute for the input field:

```
<INPUT type="text" name="usrname"
required>
```

In the preceding code snippet, the `required` attribute ensures that the text field named `usrname` should not be left blank at the time of form submission.

Defining the pattern Attribute

The `pattern` attribute is used to specify a regular expression against which the element's value will be checked. The `pattern` attribute can be used with the input types, such as `text`,`url`, `tel`, `email`, and `password`. The following table lists the expressions that can be applied on various input fields to create patterns.

Expression	***Description***
[abc]	*Finds the characters that are specified within the brackets.*
[^abc]	*Finds the characters that are not specified within the brackets.*
[0-9]	*Finds any digit from 0 to 9.*
[A-Z]	*Finds the character from uppercase A to uppercase Z.*
[a-z]	*Finds the character from lowercase a to lowercase z.*
A-z	*Finds the character from uppercase A to lowercase z.*

The Expressions that can be Applied to Create Patterns

The following code is used to specify the `pattern` attribute for the input field:

```
<INPUT type="text"
name="country_code" pattern="[A-Za-z]
{3}" title="Three letter code">
```

In the preceding code, the value for the `pattern` attribute ensures that the text field accepts only three letter alphabets in capital or small letters. It will not allow the user to enter any number or special character in the text field.

A regular expression is a set of characters, which is used to specify a pattern.

Defining the placeholder Attribute

The `placeholder` attribute is used to specify a sample value for the input field that will be displayed until the user enters a value. The `placeholder` attribute can be used with the input types, such as `text`, `tel`, `email`, `password`, and `number`. The following code snippet is used to specify the `placeholder` attribute for an input field:

```
<INPUT type="text" placeholder="Type
your first name" name="fname">
```

In the preceding code snippet, the text, `Type your First name here`, will appear in the text field named `fname.` The text will appear when the page loads suggesting the user to enter the first name in the field.

The output derived by using the `placeholder` attribute with the text field is displayed, as shown in the following figure.

The Output Derived by Using the Placeholder Attribute

Consider the following code to create input fields, such as text, date, telephone number, email, and radio for the **OrderFoodOnline** Web page:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
body{
background-color:#FFEBCD;
}
.autostyle2{
```

```
color:red;
font-size:20px;
text-align:center;
}
</STYLE>
</HEAD>
<BODY bgcolor="#FFEBCD">
<FORM action = "registeration.html">
<TABLE>
<TR class="autostyle2">
<TD colspan="4">OrderFoodOnline</
TD></TR>
<TR>
<TD WIDTH="100%" COLSPAN="4">
</TD>
</TR>
<TR>
<TD> Name:</TD>
<TD><INPUT type="text" name="usrname"
placeholder="Enter your name"
required></TD>
</TR>
<TR>
<TD>Date:</TD>
<TD><INPUT type="date" name="date"
required></TD>
</TR>
<TR>
<TD>email ID:</TD>
<TD><INPUT type="email"
name="usrmail" placeholder="Enter
your email ID" required></TD>
</TR>
<TR>
<TD>Contact Number:</TD>
<TD><INPUT type="tel" name="usrtel"
placeholder="Enter your phone number"
required></TD>
</TR>
<TR><TD>Select Food:</TD>
<TD>Non-vegetarian<INPUT type="Radio"
NAME="rd1">
Vegetarian<INPUT type="Radio"
NAME="rd1" ></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

The preceding code creates a textbox to accept the name of the customer, a date field to accept the date, an email field to accept the validated email address of the customer, a telephone field to accept the telephone number, and radio buttons to accept the food preferences of the customer. The OrderFoodOnline form is displayed, as shown in the following figure.

The OrderFoodOnline Form

<SELECT>

The <SELECT> tag is a container tag. It creates a drop-down list on the form. It has the following attributes:

- `multiple`: Is used to allow the user to select more than one value from the drop-down list by using the `Ctrl` key.
- `name`: Is used to specify a name of the selection list that will be used at the time of submitting the form.
- `size`: Is used to specify the number of visible items in the selection or drop-down list. The default value is 1. If the value of this attribute is greater than 1, then the form field will be a list.
- `autofocus`: Is used to ensure that the focus is on the drop-down list when the page loads.
- `form`: Is used to specify the name of one or more forms to which the <SELECT> tag belongs.

The following code is used to create a <SELECT> tag:

```
<SELECT name= "5-Star_Hotels" size=1
multiple></SELECT>
```

The preceding code will create a drop-down list with the name, `5-Star_Hotels`, with the size, `1`. The `multiple` attribute allows the user to select multiple items from the list.

However, the <SELECT> tag only creates a drop-down list. It does not embed list items in it. To specify the list items, you need to use the tags, <OPTION> and <OPTGROUP>, with the<SELECT> tag.

Defining the <OPTION>Tag

It is always used within the <SELECT> tag and cannot be used as a standalone tag. It is used to create a list of options in the drop-down list and has the following attributes:

- `selected`: Is used to indicate that a particular option comes pre-selected when the page loads in the browser.
- `value`: Is used to indicate the value of the option to be sent on the form submission when that option is selected by the user.
- `disabled`: Is used to indicate that an option should be disabled when the page loads.

Consider the following code to create the drop-down

lists in the **OrderFoodOnline** form:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
........</TR>
<TR>
<TD>Select Restaurant:</TD>
<TD><SELECT>
<OPTION value="opt1">Select Your
Restaurants</OPTION>
<OPTION value="opt2">La Figa</OPTION>
<OPTION value="opt3">Benihana</
OPTION>
<OPTION value="opt4">Gallipoli</
OPTION>
<OPTION value="opt5">Kings Road
SteakHouse</OPTION>
</SELECT></TD>
</TR>
<TR>
<TD>Drinks:</TD>
<TD><SELECT>
<OPTION value="opt6">Select Your
Drink</OPTION>
<OPTION value="opt7">Cappuccino</
OPTION>
<OPTION value="opt8">Caffelatte</
OPTION>
</SELECT></TD>
</TR>
<TR>
<TD> Soups:</TD>
<TD><SELECT>
<OPTION value="opt9">Select the Soup
of Your Choice</OPTION>
<OPTION value="opt10">Minestrone</
OPTION>
<OPTION value="opt11">Fonduta</
OPTION>
<OPTION value="opt12">Pasta e
fagioli</OPTION>
</SELECT></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

In the preceding code, the drop-down lists, Drinks, Soups, and Restaurant, are created. The output derived by using the `<OPTION>` tag is displayed, as shown in the following figure.

The Output Derived by Using the <OPTION> Tag

Defining the <OPTGROUP> Tag

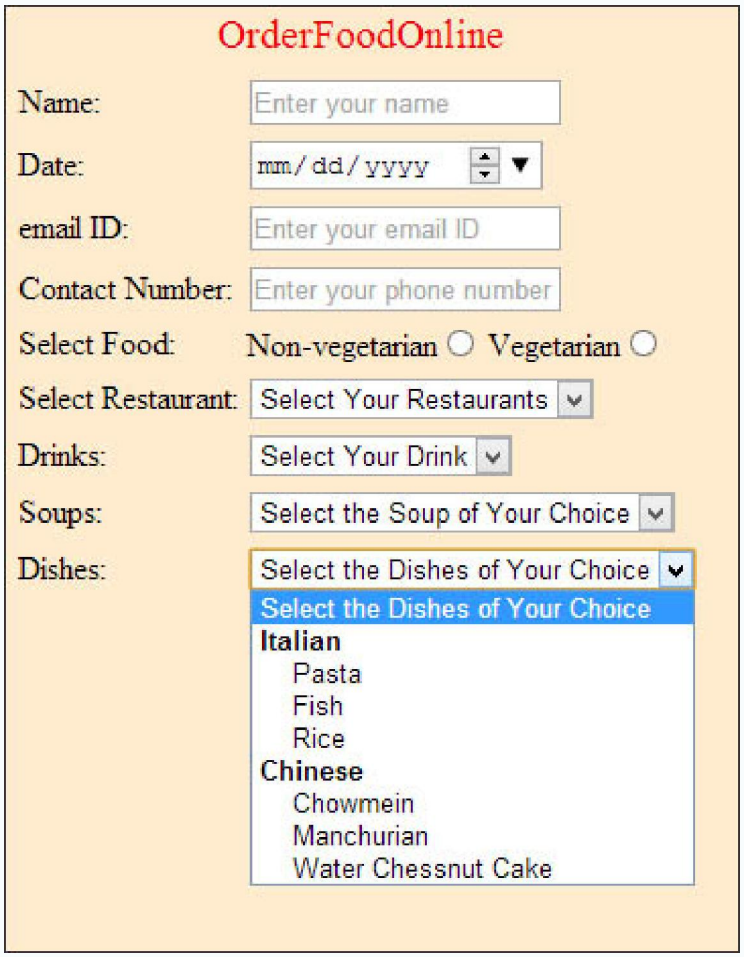
The `<OPTGROUP>` tag is used to group the related options in one group. It is generally used when you have a long list of options and you want to group the related options in one to make it simpler. The `<OPTGROUP>` tag can have the following attributes:

- `disabled`: Is used to indicate that an option group should be shown disabled when the page loads.
- `label`: Is used to specify a label for the option group.

Consider the following code to create a drop-down list in the **OrderFoodOnline** Web page:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
........</TR>
<TR>
<TD>Dishes:</TD>
<TD><SELECT>
<OPTION value="opt13">Select the
Dishes of Your Choice</OPTION>
<OPTGROUP label="Italian">
<OPTION value="opt14">Pasta</OPTION>
<OPTION value="opt15">Fish</OPTION>
<OPTION value="opt16">Rice</OPTION>
</OPTGROUP>
<OPTGROUP label="Chinese">
<OPTION value="opt17">Chowmein</
OPTION>
<OPTION value="opt18">Manchurian</
OPTION>
<OPTION value="opt19">Water Chessnut
Cake</OPTION>
</OPTGROUP>
</SELECT></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

In the preceding code, the <OPTGROUP> tag is used to group the items in two categories, Italian and Chinese. Further, the <OPTION> tag is enclosed in between the <OPTGROUP> tag to specify the items in the categories. The output derived by using the <OPTGROUP> tag is displayed, as shown in the following figure.



The Output Derived by Using the <OPTGROUP> Tag

<LABEL>

The <LABEL> tag is used to define a label for the input fields. You can also define a label for the <OUTPUT> tag. The <LABEL> element does not render anything special for the user. However, it provides functionality for users in such a way that if they click on the text within the <LABEL> element, the corresponding field will automatically get selected. For this, the for attribute of the <LABEL> tag should be equal to the ID attribute of the related input field. The <LABEL> tag has the following attributes:

- for: Is used to bind the <LABEL> tag with the input field and should have the same value as the ID attribute of the input field.
- form: Is used to specify the name of one or more forms to which the <LABEL> tag belongs.

Consider the following code to create a label for the input fields in the **OrderFoodOnline** Web page:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
body{
background-color:#FFEBCD;
}
.autostyle2{
color:red;
font-size:20px;
text-align:center;
}
</STYLE>
</HEAD>
<BODY>
  <FORM action =
"registeration.html">
<TABLE>
<TR class="autostyle2">
<TD colspan="2">OrderFoodOnline</
TD></TR>
<TR>
<TD WIDTH="100%" COLSPAN="2">
</TD>
</TR>
<TR>
<TD><LABEL for="name">Name:</LABEL></
TD>
<TD><INPUT type="text" name="usrname"
ID="name" placeholder="Enter your
name" required></TD>
</TR>
<TR>
<TD><LABEL for="date">Date:</LABEL></
TD>
<TD><INPUT type="date" name="date"
ID="date" required></TD>
</TR>
<TR>
<TD><LABEL for="email">email ID:</
LABEL></TD>
<TD><INPUT type="email"
name="usrmail" ID="email"
placeholder="Enter your email ID"
required></TD>
</TR>
<TR>
<TD><LABEL for="number">Contact
Number:</LABEL></TD>
<TD><INPUT type="tel" name="usrtel"
ID="number" placeholder="Enter your
phone number" required></TD></TR>
<TR><TD><LABEL for="food">Select
Food:</LABEL></TD>
<TD><LABEL for="nonveg">Non-
vegetarian</LABEL>
<INPUT type="radio" name="food"
ID="nonveg">
<LABEL for="veg">Vegetarian</LABEL>
<INPUT type="radio" name="food"
ID="veg"></TD>
</TR>
<TR>
<TD><LABEL for="restro">Select
```

```
Restaurant:</LABEL></TD>
<TD><SELECT>
<OPTION value="opt1">Select Your
Restaurants</OPTION>
   <OPTION value="opt2">La Figa</
OPTION>
   <OPTION value="opt3">Benihana</
OPTION>
   <OPTION value="opt4">Gallipoli</
OPTION>
     <OPTION value="opt5">Kings Road
SteakHouse</OPTION>
</SELECT></TD>
</TR>
<TR>
<TD><LABEL for="drinks">Drinks:</
LABEL></TD>
<TD><SELECT>
<OPTION value="opt6">Select Your
Drink</OPTION>
   <OPTION value="opt7">Cappuccino</
OPTION>
   <OPTION value="opt8">Caffelatte</
OPTION>
</SELECT></TD>
</TR>
<TR>
<TD><LABEL for="soups">Soups:</
LABEL></TD>
<TD><SELECT>
<OPTION value="opt9">Select the Soup
of Your Choice</OPTION>
   <OPTION value="opt10">Minestrone</
OPTION>
     <OPTION value="opt11">Fonduta</
OPTION>
  <OPTION value="opt12">Pasta e
fagioli</OPTION>
</SELECT></TD>
</TR>
<TR>
<TD><LABEL for="dishes">Dishes:</
LABEL></TD>
<TD><SELECT>
<OPTION value="opt13">Select the
Dishes of Your Choice</OPTION>
<OPTGROUP label="Italian">
   <OPTION value="opt14">Pasta</
OPTION>
   <OPTION value="opt15">Fish</
OPTION>
  <OPTION value="opt16">Rice</OPTION>
</OPTGROUP>
<OPTGROUP label="Chinese">
   <OPTION value="opt17">Chowmin</
OPTION>
     <OPTION
value="opt18">Manchurian</OPTION>
  <OPTION value="opt19">Water
Chessnut Cake</OPTION>
</OPTGROUP>
</SELECT></TD>
</TR>
<TR>
<TD><LABEL for="Order For:">Order
For:</LABEL></TD>
<TD><LABEL for="pickup">Take Away</
LABEL>
<INPUT type="radio" name="pickup"
ID="pickup">
<LABEL for="home">Home Delivery</
LABEL>
<INPUT type="radio" name="pickup"
ID="home"></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

In the preceding code, the <LABEL> tag is used to label the input fields in the **OrderFoodOnline** Web page. You can select the input field by clicking on the label of the field. The output derived by using the <LABEL> tag is displayed, as shown in the following figure.

The Output Derived by Using the <LABEL> Tag

<FIELDSET>

The <FIELDSET> tag is used to combine and group related fields in a form. It creates a box around the selected fields. You can also define the description for the fieldset by using the <LEGEND> tag. The <LEGEND> tag is used along with the <FIELDSET> tag to define a caption for the fieldset. It is the simplest way of organizing form elements along with their description in such a way that it is easier for a user to understand.

The <FIELDSET> tag can have the following attributes:

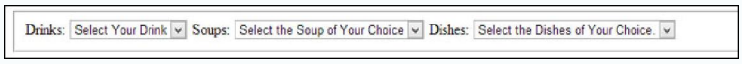
- q disabled: The disabled attribute is used to indicate that a group of fields should be shown disabled when the page loads.
- q form: The form attribute is used to specify the name of one or more forms to which the <FIELDSET> tag belongs.
- q name: Thename attribute is used to specify the name for the fieldset.

Consider the following code to group the drop-down lists inside a fieldset on the **OrderFoodOnline** Web page:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<FORM>
<FIELDSET>
Drinks:
<SELECT>
  <OPTION value="opt1">Select Your
Drink</OPTION>
<OPTION value="opt2">Cappuccino</
OPTION>
<OPTION value="opt3">Caffelatte</
OPTION>
</SELECT>
Soups:
<SELECT>
<OPTION value="opt4">Select the Soup
of Your Choice</OPTION>
<OPTION value="opt5">Minestrone</
OPTION>
  <OPTION value="opt6">Fonduta</
OPTION>
<OPTION value="opt7">Pasta e
fagioli</OPTION>
</SELECT>
Dishes:
<SELECT>
<OPTION value="opt8">Select the
Dishes of Your Choice.</OPTION>
<OPTGROUP label="Italian">
<OPTION value="opt9">Pasta</OPTION>
  <OPTION value="opt10">Fish</OPTION>
<OPTION value="opt11">Rice</OPTION>
</OPTGROUP>
<OPTGROUP label="Chinese">
<OPTION value="opt12">Chowmin</
OPTION>
   <OPTION value="opt13">Manchurian</
OPTION>
<OPTION value="opt14">Water Chessnut
Cake</OPTION>
</OPTGROUP>
</SELECT>
</FIELDSET>
</FORM>
</BODY>
</HTML>
```

The preceding code groups the food items in one box.

The output derived by using the <FIELDSET> tag is displayed, as shown in the following figure.



The Output Derived by Using the <FIELDSET> Tag

<TEXTAREA>

The <TEXTAREA> tag creates a field in which the user can enter a large amount of text. The <TEXTAREA> tag has the following attributes:

- q rows
- q cols

rows

The rows attribute helps to set the number of rows of text that will be visible without scrolling up or down in the field.

cols

The cols attribute helps to set the number of columns of text that will be visible without scrolling right or left in the field.

Consider the following code to create a textarea in the **OrderFoodOnline** Web page to accept the users' address:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
.........</TR>
<TR>
<TD><LABEL for="Orderfor">Adderess:</
LABEL></TD>
<TD><TEXTAREA rows="3" cols="16"
ID="Orderfor">
Enter Your Address Here
</TEXTAREA></TD>
</TR></TABLE>
</FORM>
</BODY>
</HTML>
```

The output derived by using the <TEXTAREA> tag is displayed in the following figure.

The Output Derived by Using the <TEXTAREA> Tag

<DATALIST>

The <DATALIST> tag is used to create a list of pre-defined options for an input field. It is used to provide an autocomplete feature on input fields so that the user can view the drop-down list of pre-defined options whenever they input data.

Consider the following code to create a datalist named **State** in the **OrderFoodOnline** Web page:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
........</TR>
<TR>
<TD><LABEL for="state">State:</
LABEL></TD>
<TD><INPUT list="stat" name="stat"
ID="state">
<DATALIST ID="stat">
  <OPTION value="Alabama">
  <OPTION value="California">
  <OPTION value="Delaware">
  <OPTION value="Florida">
  <OPTION value="Hawaii">
</DATALIST><BR>
</TD>
</TR> </TABLE>
</FORM>
</BODY>
</HTML>
```

The preceding code creates a datalist so that whenever a user types the initial characters of the name of a state, the names matching the initials will appear in the drop-down list, as shown in the following figure.

The Output Derived by Using the <DATALIST> Tag

In the preceding figure, when a user types the character, c, California is prompted in the list.

<KEYGEN>

The <KEYGEN> tag is used to specify a key-pair generated field in a form. Whenever the form is submitted, the private and public keys are generated, where the private key is stored locally, and the public key is sent to the server. As the public key is stored in the server, it can be used to authenticate a user in the future. The <KEYGEN> tag can have the following attributes:

- autofocus: Is used to specify that the <KEYGEN> tag automatically gets the focus when a Web page loads.
- disabled: Is used to indicate that a <KEYGEN> tag should be shown disabled when a page loads.
- name: Is used to specify a name for the <KEYGEN> tag.
- keytype: Is used to specify the security algorithm of the key. It accepts the name of various security algorithms, such as rsa, dsa, andec, as its value.
- form: Is used to specify the name of one or more forms to which the <KEYGEN> tag belongs.

The rsa, dsa, and ec are the different security algorithms that are used for public key encryption.

The <KEYGEN> tag can be created using the following

code:

```
<KEYGEN name="key1" keytype="rsa">
```

The preceding code creates a key-pair generated field using the security algorithm, `rsa`.

<OUTPUT>

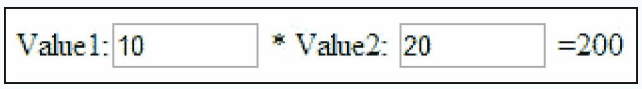
The `<OUTPUT>` tag is used to represent the result of a calculation. The `<OUTPUT>` tag can have the following attributes:

- `for`: Is used to specify the relationship between the input fields used and the result generated for the calculation.
- `form`: Is used to specify the name of one or more forms to which the `<OUTPUT>` tag belongs.
- `name`: Is used to specify a name for the `<OUTPUT>` tag.

Consider the following code to get the result of the multiplication of two numbers by using the `<OUTPUT>` tag:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<FORM ONINPUT="mul.value=parseInt
(val1.value)*parseInt(val2.value)">
Value1:<INPUT type="text"
name="val1">
* Value2: <INPUT type="text"
name="val2">
=<OUTPUT name="mul" for="val1
val2"></OUTPUT>
</FORM>
</BODY>
</HTML>
```

The preceding code creates two input fields, `val1` and `val2`, to accept user input. Further, the `ONINPUT` event executes on the form and multiplies the values present in the input fields, `val1` and `val2`. Then, the `<OUTPUT>` tag is used to display the result of the calculation. The output derived by using the `<OUTPUT>` tag is displayed, as shown in the following figure.



The Output Derived by Using the <OUTPUT> Tag

<BUTTON>

The `<BUTTON>` tag is used to create a button. However, unlike the input type, `submit`, you can specify a text or an image within the button by using the `<P>` or `<IMG>` tags. The `<BUTTON>` tag has the following attributes:

- `type`: Is used to specify the type of the button. It can accept the values, such as `button`, `submit`, and `reset`. The `button` value creates a simple push button, the `submit` value creates a button that submits the form data, and the `reset` value creates a button that reset the form fields.
- `name`: Is used to specify the name of the button.
- `form`: Is used to specify the name of one or more forms to which the button belongs.
- `autofocus`: Is used to specify that the button automatically gets the focus when the Web page loads.
- `disabled`: Is used to indicate that the button should be shown disabled when the Web page loads.

Consider the following code to create buttons, Submit and Reset, in the **OrderFoodOnline** Web page.

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
#button{
Margin-left:50px;
}
.....
/HEAD
BODY
........</TR>
<TR>
<TD>
<BUTTON ID="button"
type="submit"><P>Submit</P></
BUTTON></TD><TD>
<BUTTON type="reset"><P>Reset</P></
BUTTON>
</TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

The preceding code creates two buttons, Submit and Reset, as displayed in the following figure.

OrderFoodOnline

Name: Enter your name

Date: mm/dd/yyyy

email ID: Enter your email ID

Contact Number: Enter your phone number

Select Food: Non-vegetarian ○ Vegetarian ○

Select Restaurant: Select Your Restaurants

Drinks: Select Your Drink

Soups: Select the Soup of Your Choice

Dishes: Select the Dishes of Your Choice

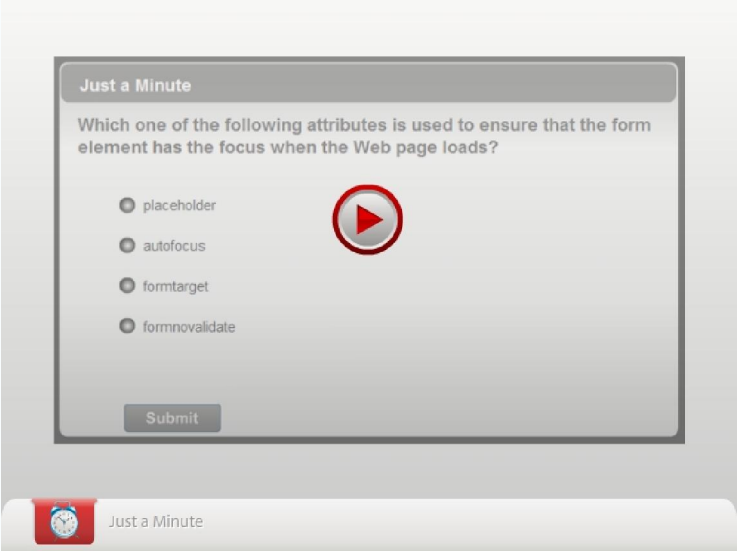
Order For: Take Away ○ Home Delivery ○

Address: Enter Your Address Here

State:

Submit Reset

The Output Derived by using the < BUTTON> Tag



Activity 5.1: Designing an HTML Form

Manipulating the Components of a Web Page

The website of BookYourHotel.com enables the customers to access the online booking form for each hotel by clicking on the image of the hotel. In addition, a clock is displayed on the Web page, which is dynamically updated every second to display the current time to the users.

To implement such functionalities on the Web pages, JavaScript is used. It is an object-based language and treats every element, such as an image or a button inside a browser window as an object. Every object has a pre-defined set of attributes and events associated with it. JavaScript also provides various types of built-in objects, such as browser objects and form objects, which help to make a Web page dynamic and interactive.

Working with Browser Objects

When a Web page is displayed in a browser, a user can access information not only about the current page but also additional details, such as information about the window and screen, the pages the user has visited in the past, or the version of the browser being used to view the document.

All this information is accessible by using browser objects. In addition, the browser parses the currently displayed Web page into multiple objects, such as a window, screen, and a document. The objects that define the browser content and the browser itself are known as browser objects. Browser objects enable retrieval and manipulation of information about a browser, such as the window size, height, width, and name. These objects also enable access to information, such as browsing history and current version of a browser.

JavaScript defines the following browser objects on a Web page:

- `window`
- `document`
- `navigator`
- `screen`
- `history`
- `location`

Browser objects represent the browser environment and provide properties and methods for its access and manipulation. The browser environment refers to components, such as the window in which the document is displayed and the history list that contains information regarding the Web pages visited by a user. You can set the size, name, and default status of the browser window by using the window object. You can also manipulate URL of the document using the `location` object. Similarly, you can access the browser version and the browser name in which the Web page is displayed using the `navigator` object. This information can be useful for a correct display of a Web page in the browser.

Using the window Object

The `window` object is one of the highest-level objects in the JavaScript object hierarchy. It represents a browser window, which displays the document. It can also be a combination of multiple frames. Each frame within a window is itself a window. The following table lists

some of the properties of the window object.

Properties	Description	Syntax
defaultStatus	Is a string value containing the default text that appears on the status bar of the window.	window.defaultStatus
document	Is a reference to the document displayed in the window.	window.document
frames[]	Is an array that represents all the frames in the window. You can refer to a particular frame by using the frames[] property.	window.frames[i], where i is the index of a particular frame in a window.
frames.length	Is an integer value representing the number of frames in the window.	window.frames.length
name	Returns or sets a name for the window.	window.name
parent	Returns the parent window of the current window.	window.parent
self	Returns the current window.	window.self
top	Returns the topmost browser window. The topmost window is the current active window overlapping all the open windows.	window.top
status	Is a string value and is used to set the text on the status bar of the window.	window.status

The Properties of the window Object

The following table lists some of the methods of the window object.

Methods	Description	Syntax	Example
open()	Opens a new browser window.	window.open(URL, Name, specs, replace); where, URL is the address of the Web page to be displayed. Name is the name of the window in which the Web page is to be displayed. specs define the specifications of the window, such as its height and width, in the form of a comma-separated list of attributes. replace is used to determine whether to replace the current document or create a new entry in the history list using true or false values, respectively.	window.open ("index.html", "Home", "height=100,width=200", false); where, the URL is index.html. The Name of the window is Home. The height and width of the screen is 100 and 200, respectively.
close()	Closes the current window.	window.close();	window.close() Closes the current window.
alert()	Displays a message in a dialog box.	window.alert(messagetext); where, messagetext is the text to be displayed in the dialog box.	window.alert("Hello"); Displays the message Hello in a dialog box.
setTimeout()	Calls a function after a specified number of milliseconds.	window.setTimeout(function, time)	var t=setTimeout ("alertMessage()",3000); Calls the alertMessage() function after 3 seconds.

clearTimeout()	Cancels a timer that is set using the setTimeout() method. It takes the timerID parameter returned by the setTimeout() method.	window. clearTimeout(timerID)	t=setTimeout ("timedCount()",1000);
			clearTimeout(t);
			where, t is the timerID parameter returned by the setTimeOut() method.

The Methods of the window Object

The window *object is a default object when writing the JavaScript code. Therefore, it is not necessary to explicitly qualify the methods and properties of the* window *object using the dot operator.*

Consider the following code snippet for opening a new window displaying the hotel bookings form, when a user clicks the hotel image:

```
function open_win()
{
window.open
("HotelBooking1.html","height=100,wid
th=200");
}
```

In the preceding code snippet, when the function, open_win(), is executed, it opens the **purchase.html** Web page in a new window.

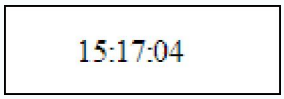
Consider the following code to display the clock on the Web page of the BookYourHotel.com website by using the setTimeout() method:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
function clockTime()
{
var todayDate=new Date();
var hrs=todayDate.getHours();
var mns=todayDate.getMinutes();
var scs=todayDate.getSeconds();
mns=check(mns);
scs=check(scs);
document.getElementById
```

```
('displayTime').innerHTML=hrs+":"+mns
+":"+scs;
t=setTimeout('clockTime()',1000);
}
function check(t)
{
if (t<10)
{
t="0" + t;
}
return t;
}
</SCRIPT>
</HEAD>
<BODY onload="clockTime()">
<DIV ID="displayTime"></DIV>
</BODY>
</HTML>
```

In the preceding code, the `clockTime()` function is called to display a clock on the Web page. The `clockTime()` function uses the `Date` object to get the current time in hours, minutes, and seconds. It updates time every second using the `setTimeout()` function. The `setTimeout()` function calls the `clockTime()` function after every second or 1000 milliseconds. The `clockTime()` function uses the `check(t)` function to check if the number of minutes or seconds is less than 10. If this is true, 0 is displayed before the number of minutes or seconds. For example, if the current time is 14 hours 45 minutes and 3 seconds, the time is displayed as 14:45:03.

The output of the preceding code to display a clock on a Web page is displayed, as shown in the following figure.



The Clock Displayed on a Web Page

Using the document Object

The `document` object is subordinate to the `window` object in the document object model hierarchy. The `document` object provides the properties and methods to work with many aspects of the current document, including information about anchors, forms, links, title, current location and URL, and the current colors. The HTML document that is loaded on the Web browser window acts as a `document` object.

The following table lists some of the properties of the `document` object.

Properties	***Description***
`alinkColor`	*Is a string value representing the color of an active link.*
`anchors[]`	*Is an array object containing references to all the anchor elements in a document.*
`bgColor`	*Is a string value representing the background color of the document.*
`cookie`	*Is a string value containing name/value pairs of data that will persist in the memory of the client computer until the Web browser is cleared or the expiry date is reached.*
`fgColor`	*Is a string value representing the text color of the document.*
`forms[]`	*Is an array object containing references to each form in the document. Form elements are contained within the form object.*
`linkColor`	*Is a string value representing the color of an unvisited link.*
`lastModified`	*Is a string value representing the date and time when the document was last modified.*
`links[]`	*Is an array object containing references of all the elements in the <A> tag and elements that use the <AREA> tag.*
`referrer`	*Is a string value representing the URL of the document from which the current document is accessed.*
`title`	*Is a string value representing the title of the document.*
`vlinkColor`	*Is a string value representing the color of the visited link.*

The Properties of the document Object

Some of the widely used methods of the `document` object are:

- `write()`
- `writeln()`
- `getElementsByName()`
- `getElementsByTagName()`
- `getElementById()`

write()

The `document.write()` method enables users to write text on a Web page. The following syntax is used to write the text , Hello!, using the `document.write`

()method on a Web page:

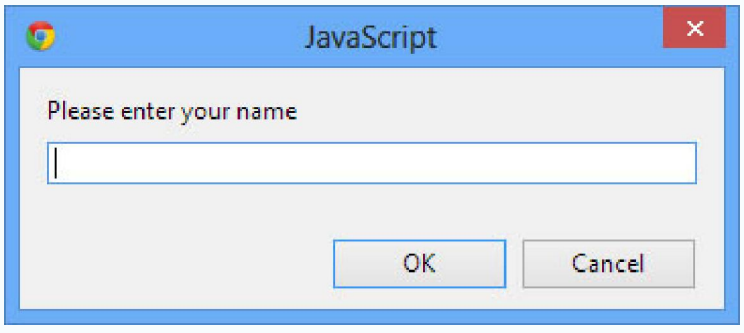
```
document.write("Hello!");
```

You can also add HTML elements to a Web page dynamically using the `document.write()` method.

For example, you can accept a name as input and then use the `document.write()` method to write it on a Web page. You can even dynamically specify the formatting of the content on the Web page, as shown in the following code:

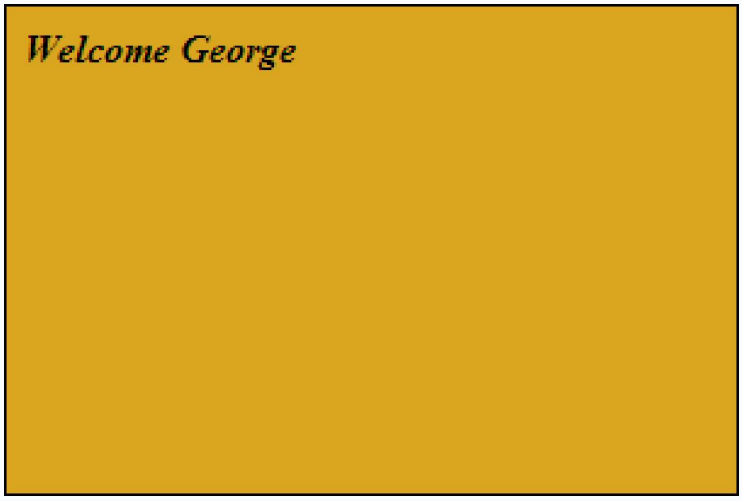
```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<TITLE>JavaScript Write method
Illustration</TITLE>
<STYLE>
body{
background-color:#DAA520;
}
</STYLE>
</HEAD>
<BODY>
<SCRIPT type="text/javascript">
{
var name= prompt("Please enter your
name","");
document.write("<P>");
document.write("<I>");
document.write("<B>");
document.write("Welcome "+name);
document.write("</B>");
document.write("</I>");
document.write("</P>");
}
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the `prompt()` method is used to prompt the user to enter his/her name, as shown in the following figure.



The Prompt Box

Assuming that the user enters the name, George, in the prompt box, the output is displayed as shown in the following figure.



The Output of Code Using the write() Method

In the preceding code, the `document.write()` method is used in the `<SCRIPT>` tag to display the welcome message on the Web page.

writeln()

The `writeln()` method also writes text on a Web page. The only difference is that the `writeln()` method appends a carriage return character to the end of the output. The carriage return character is used to write data on separate lines of the Web page. For example, consider the following code snippet:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<PRE>
<SCRIPT type="text/javascript">
document.writeln("Hi!");
document.writeln("Welcome to our
site!");
document.write("Have a ");
document.write("great day");
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

The `writeln()` method appends a newline character at the end of the text. However, HTML ignores newline when writing the output text. The `<PRE>` tag can be used to display the pre-formatted output on a Web page. When the `<SCRIPT>` tag is enclosed within the `<PRE>` tag, the newline appended using the `document.writeln()` method is rendered.

When the preceding code snippet is executed, the message, **Hi!**, is displayed, and the carriage return character is added at the end of this message so that the next message is displayed in a new line.

The output of the preceding code snippet is displayed, as shown in the following figure.

```
Hi!
Welcome to our site!
Have a great day
```

The Output Derived Using the writeln() Method

getElementsByName()

It is recommended that each element on a Web page should be specified by a unique name. However, multiple elements can also share the same name in a Web page. The `document.getElementsByName()` method is used to access all the elements with the specified name used in an HTML document. This method returns an array of all the elements with the specified name in the HTML document. The following syntax is used for the `getElementsByName()` method:

```
document.getElementsByName
("name_of_the_element");
```

In the preceding syntax, `name_of_the_element`, specifies the name of the element(s) to be accessed.

Consider the following code:

```
<!DOCTYPE HTML>
<HTML>
<SCRIPT type="text/javascript">
function count()
{
var x=document.getElementsByName
("link");
alert(x.length + "Hyperlinks");
}
</SCRIPT>
<BODY>
<A name="link" href="" >Link 1</
A><BR/>
<A name="link" href="" >Link 2</
A><BR/>
<A name="link" href="" >Link 3</
A><BR/>
<BR/>
<INPUT type="button" value="Count"
onclick="count()"/>
</BODY>
</HTML>
```

In the preceding code, the `length` property is used to count the occurrences of all the elements that have their name specified as **link**. Therefore, the statement `x.length` will generate the value, 3, and display it in an alert box.

getElementsByTagName()

Consider a scenario, where a user can customize the background color of all the text boxes on a Web page. A drop-down list is used to choose the background color. If the user selects the green color from a drop-down list, the background color of all the text boxes in the document changes to green. This can be achieved using the `document.getElementsByTagName()` method. The `document.getElementsByTagName()` method is used to access all the elements of the same type in the Web page. In the given scenario, the `document.getElementsByTagName()` method can be used to return an array of all the text boxes in the document which can then be manipulated using CSS to achieve the desired functionality. The following syntax is used for the `getElementsByTagName()` method:

```
document.getElementsByTagName
("Tag_name");
```

In the preceding syntax, `Tag_name` specifies the tag name of the element(s) that have to be accessed.

Consider the following code snippet:

```
var x=document.getElementsByTagName
("a");
alert(x.length + "Hyperlinks");
```

In the preceding code snippet, the number of anchor or `<A>` tags on the Web page is counted and displayed in an alert box. For example, if there are four `<A>` tags in a document, then the output of the preceding code will be `4 Hyperlinks`.

getElementById()

As you know, each HTML element used on a Web page has an optional attribute, `ID`, which uniquely identifies the element. The `document.getElementById()` method uses the ID of HTML elements to access and manipulate their content.

The following syntax is used to access an element using the `getElementById()` method:

```
document.getElementById
("id_of_the_element");
```

In the preceding syntax, `id_of_the_element`, specifies the ID of the element that has to be accessed.

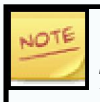
Consider the following code snippet to use a text box within the Web page:

```
<INPUT type="text" ID="text1"/>
```

To access the value of the text box inside the script, you need to use the following code snippet:

```
var name = document.getElementById
("text1").value;
```

In the preceding code snippet, the value of the text box is fetched using `ID` of the text box specified as `text1`. The value is then stored in the `name` variable.



The `value` property is used to retrieve the value of the elements that are specified using the `<INPUT>` tag.

Using JavaScript, you can also change the content of the HTML elements, such as paragraphs, hyperlinks, and headers. To retrieve and change the text of the HTML elements that have both a start tag and an end tag, such as `<P>`, `<OPTION>`, and `<DIV>`, you can make use of the `innerHTML` property with the `document.getElementById()` method. Consider the following code snippet:

```
<!DOCTYPE HTML>
<HTML>
```

```
<SCRIPT type="text/javascript">
function replacetext()
{
document.getElementById
("para1").innerHTML="Changed the
content by using the innerHTML
property of getElementById() method";
}
</SCRIPT> <BODY>
<P ID="para1">A simple paragraph</P>
<INPUT type="button" value="Change
text" onclick="replacetext()"/> </
BODY>
</HTML>
```

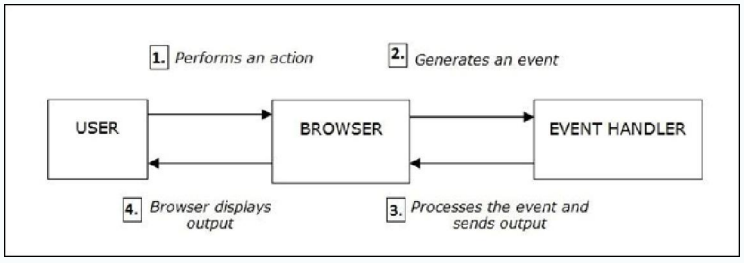
In the preceding code snippet, when the **Change text** button is clicked, the `replacetext()` function is called using the `onclick` event. The `replacetext()` function replaces the text of the `<P>` element with the text, **Changed the content by using the innerHTML property of getElementById() method**.

Events are discussed in the next topic of this section.

Handling Events

In JavaScript, an event is an action that happens on a Web page, such as a mouse click or loading of a Web page. The Web browser waits for the event to occur, and when it occurs, performs the processing that has been programmed for the event. For example, when a user clicks the **OK** button on a form, the Web browser executes a part of the code, which is designed to execute on that event. This is called event handling. The function, which is executed in response to an event, is called an event handler.

The following figure displays the event handling process.



The Event Handling Process

The preceding figure depicts the process where a user interacts with the browser and an event is generated. This event is processed by the event handler and the output is sent to the browser. Further, the browser displays the output to the user.

The following table lists the various types of events supported in JavaScript.

Event	*Description*	*Example*
`onblur`	*Occurs when an element loses focus.*	`<INPUT type="text" name="textbox1" onblur="show()"/>` *where,* `show()` *is the function to be executed when the event occurs.*
`onchange`	*Occurs when the content of a field changes.*	`<INPUT type="text" ID="fname" onchange="show()" />` *where,* `show()` *is the function to be executed when the event occurs.*
`onclick`	*Occurs when the user clicks an object.*	`<BUTTON onclick="show() ">Show Text</ BUTTON>` *where,* `show()` *is the function to be executed when the event occurs.*
`onfocus`	*Occurs when an element gets focus.*	`<INPUT type="text" ID="firstname" onfocus="alert ('hello')"; />`
`onkeydown`	*Occurs when any key of the keyboard is pressed.*	`<INPUT type="text" onkeydown="alert('key is down')"/>`
`onkeypress`	*Occurs when a character key of the keyboard is pressed or held down.*	`<INPUT type="text" onkeypress="alert('key is pressed')"/>`
`onkeyup`	*Occurs when a keyboard key is released.*	`<INPUT type="text" onkeyup="alert('key is released')"/ >`
`onload`	*Occurs when a page or image has finished loading.*	`<BODY onload="load()">` *where,* `load()` *is the function to be executed when the event occurs.*
`onmousedown`	*Occurs when the*	`<IMG`

	mouse button is pressed.	*src="image11.gif" onmousedown="alert('You clicked image11!')">*
onmousemove	*Occurs when the mouse is moved.*	*<IMG src="image11.gif" onmousemove="alert('You moved your mouse over image11!')">*
onmouseout	*Occurs when the mouse is moved off an element.*	*<P onmouseout="alert('Mouse moved out of the paragraph')"> Move the mouse pointer out of my paragraph to display an alert box.</P>*
onmouseover	*Occurs when the mouse moved over an element.*	*<P onmouseover="alert(' moved over the paragraph')"> Move the mouse pointer over me to display an alert box.</P>*
onmouseup	*Occurs when the mouse button is released.*	*<IMG src="image11.gif" onmouseup="alert('You clicked image11')" />*
onselect	*Occurs when a text is selected.*	*Select text: <INPUT type="text" value="Select me!" onselect="alert('You have selected the text.')" />*
onunload	*Occurs when the user exits the page.*	*<BODY onunload="alert('The page is unloaded')">*
ondblclick	*Occurs when the user double-clicks an object.*	*<BUTTON ondblclick="show()">Show Text</BUTTON> where, show() is the function to be executed when the event occurs.*
onerror	*Occurs when an error occurs while loading a document or an image.*	*<IMG src="image1.jpg" onerror="alert('Cannot load image.')">*

The Events Supported in JavaScript

Consider the scenario of BookYourHotel.com website. The user registration page displays the registration form, which a user should fill to get registered on the website. The management wants to implement the following functionality on the registration page:

- If user leaves any text box blank, its background color should change to pink, otherwise, it should remain white.
- After the user fills the required details in the user registration form and clicks the **Register me** button, a confirm box should be displayed highlighting the registration information entered by the user.

Consider the following code to implement these functionalities:

```
<!DOCTYPE HTML>
<HTML> <HEAD>
<TITLE>REGISTRATION DETAILS</TITLE>
<STYLE>
h1,h3{
color: black;
font-size: 40px;
text-align:center;
}
table{
margin-left:650px;
border:2px solid white;
background-color:beige;
}
td{
padding:10px;
border:2px solid white;}
#button{
margin-left:740px;}
```

```
#button{
margin-left:740px;}
</STYLE>
<SCRIPT>
function show()
{
var fname = document.getElementById
("txtbox1").value;
var lname = document.getElementById
("txtbox2").value;
var age = document.getElementById
("age").value;
var address = document.getElementById
("address").value;
var gender=document.getElementById
("gender").value;
confirm("You have entered:" + "\n
Name : " + fname + " " + lname + "\n
Age : " + age + "\n Address : "+
address + "\n Gender : "+ gender +
"\n\n Do you want to confirm these
details ?");
}
function changeColor(val)
{
if((val.value=="")||
(val.value==null))
{
val.style.background="pink";
}
else
{
val.style.background="#FFFFFF";
}
}
</SCRIPT>
</HEAD>
<BODY>
<H1>USER REGISTRATION DETAILS<HR/></
H1>
<H3>Please fill the following details
and get registered !!</H3><BR/><BR/>
<TABLE>
<TR>
<TD> First name: </TD>
<TD> <INPUT type="text" name="name1"
ID="txtbox1" onblur="changeColor
(this)"/> </TD>
</TR> <TR>
<TD> Last name: </TD>
<TD><INPUT type="text" name="name2"
ID="txtbox2" onblur="changeColor
(this)"/></TD>
</TR> <TR>
<TD> Age: </TD>
<TD> <INPUT type="text"
name="age_box" ID="age"
onblur="changeColor(this)"/> </TD>
</TR> <TR>
<TD> Address:</TD>
<TD><TEXTAREA rows="5"
name="address_box" ID="address"
onblur="changeColor(this)"></
textarea></TD>
</TR> <TR>
<TD> Gender: </TD> <TD>
<SELECT name="Gender" ID="gender">
<OPTION value="Male">Male</OPTION>
<OPTION value="Female">Female</
option> </SELECT> </TD>
</TR> </TABLE>
<BR/><INPUT ID="button" type="button"
value="Register Me" onclick="show()"/
>
</BODY>
</HTML>
```

The keyword, `this`, refers to the current objects, such as the text box and button, in the document.

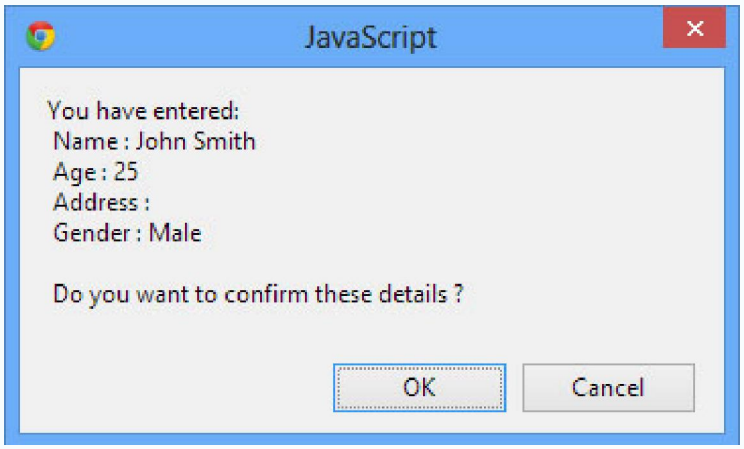
The Web page for the preceding code is displayed in the following figure.

The Browser Output

The preceding output allows the user to enter details in the registration form. As the user fills a text box and presses the **Tab** key to move to the next one, the background color of the unfilled text box changes to yellow. The following figure displays the completed User Registration form.

The Completed User Registration Form

When the user clicks the **Register Me** button, a confirm dialog box is displayed, as shown in the following figure.



A Confirm Dialog Box

Event Listeners

In JavaScript, you can handle user actions or events by calling specific methods known as event handlers. To handle each and every user action or event, a listener is notified. A listener is an object that waits for an event to occur and performs certain actions corresponding to it. However, for this, you need to register it with the event handler. The event handler function registered with the event of the field processes the event and provides an appropriate response to the listener. Then, the action is performed by the event listener.

To register a handler, you need to use the `addEventListener()` function. The function receives an event object that encapsulates the event information. The following syntax can be used to create an `addEventListener()` function:

```
addEventListener(type, listener[, useCapture]);
```

In the preceding syntax:

- `type`: Specifies a string representing the event type.
- `listener`: Specifies the object that receives a notification when an event occurs.
- `useCapture`: Is a Boolean variable, specifying whether an event needs to be captured or not. It is an optional parameter.

If the event handler captures an event, every time when the event occurs on the element, the event handler will be called. You can also remove a handler by calling the `removeEventListener()` function. However, the parameters passed to this function must be identical to the parameters passed to the `addEventListener()` function.

Consider the following code to understand the functionality of event listeners:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
function OnClick () {
alert ("A click event has occurred on the Submit button.");
}
function AddEventHandler () {
var Button = document.getElementById ("Button");
if (Button.addEventListener) {
Button.addEventListener ("click", OnClick, false);
}
}
function RemoveEventHandler () {
var Button = document.getElementById ("Button");
if (Button.removeEventListener) {
Button.removeEventListener ("click", OnClick, false);
}
}
</SCRIPT>
</HEAD>
<BODY>
Click on the Submit button when the 'click' event has a listener and when it does not.<BR />
<BUTTON onclick="AddEventHandler ();">Add a 'click' event listener to the Submit button</BUTTON>
<BUTTON onclick="RemoveEventHandler ();">Remove the event listener</BUTTON>
<BR /><BR />
<BUTTON ID="Button">Submit</BUTTON>
</BODY>
</HTML>
```

Using the navigator Object

The `navigator` object is one of the top-level objects in the JavaScript object hierarchy. Unlike the `window` or `document` objects, the `navigator` object is an independent object with its own set of properties and methods. An independent object does not have any child objects in its hierarchy.

You can use the `navigator` object to access information about the current browser, such as the browser name, version, and the user platform.

The following table lists the properties of the `navigator` object.

Properties	***Description***	***Syntax***
appName	*Specifies the name of the Web browser.*	*navigator.appName*
appVersion	*Specifies information about the Web browser version, browser platform, and the*	*navigator.appVersion*

	country for which the Web browser is released.	
`appCodeName`	*Specifies the internal code name of the Web browser. For example, the internal code name returned for both, Microsoft Internet Explorer and Netscape browsers, is Mozilla.*	`navigator.appCodeName`
`userAgent`	*Sends a string containing information, such as browser version, code name, and platform from the client to the server. The server uses this string to identify the client.*	`navigator.userAgent`
`platform`	*Specifies the operating system of the client machine.*	`navigator.platform`

The Properties of the navigator Object

The following table lists the methods associated with the `navigator` object.

Methods	***Description***	***Syntax***
`javaEnabled()`	*Returns a Boolean value that specifies whether JavaScript is enabled or disabled in the Web browser. For example, the value True indicates that JavaScript is enabled in the current Web browser.*	`navigator.javaEnabled()`
`taintEnabled()`	*Returns a Boolean value that specifies whether the security feature of the current Web browser is enabled or disabled. By default, the value returned is* `False`*.*	`navigator.taintEnabled()`

The Methods of the navigator Object

Consider the following code to use the `navigator` object:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
document.write(navigator.appName);
document.write(navigator.appVersion);
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the `appName` and `appVersion` properties of the `navigator` object are used to display the name and version of the browser, respectively.

Using the screen Object

The `screen` object enables you to access details of the user's screen, such as its width, height, and resolution.

The following table lists the most commonly used properties of the `screen` object.

Properties	***Description***	***Syntax***
`height`	*Returns the total height of the screen.*	`screen.height`
`pixelDepth`	*Returns the color resolution of the screen.*	`screen.pixelDepth`
`width`	*Returns the width of the screen.*	`screen.width`
`availHeight`	*Returns the height of the screen without including the Windows taskbar.*	`screen.availHeight`
`availWidth`	*Returns the width of the screen without including the Windows taskbar.*	`screen.availWidth`
`colorDepth`	*Returns the bit depth of color palette that is used for displaying images.*	`screen.colorDepth`

The Properties of the screen Object

Consider the following code that illustrates the use of the `screen` object:

```
<!DOCTYPE HTML>
```

```
<HTML>
<BODY>
<SCRIPT type="text/javascript">
document.write("Height: " +
screen.availHeight);
document.write("Width: " +
screen.availWidth);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code displays the height and width of the screen excluding the Window taskbar using the `availHeight` and `availWidth` properties.

Using the history Object

The `history` object contains a list of all the pages that have been visited by the user.

The following table lists the properties of the `history` object.

Properties	***Description***	***Syntax***
`length`	*Is an integer value representing the number of links currently referenced by the* `history` *object in the current session.*	*`history.length`*

The Properties of the history Object

The following table lists the most commonly used methods of the `history` object.

Methods	***Description***	***Syntax***
`back()`	*Is used to move to the previous page.*	*`history.back()`*
`forward()`	*Is used to move to the next page.*	*`history.forward()`*
`go()`	*Is used to move back specific number of pages.*	*`history.go(x)`* *where* `x` *is the number of pages*

The Methods of the history Object

Consider the following code to use the `history` object:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
function show()
{
alert(history.length)
}
</SCRIPT>
</HEAD>
<BODY>
<INPUT type="button" value="Show
History" onclick="show()" />
</BODY>
</HTML>
```

The preceding code displays the number of links referenced in the current browser session when a user clicks the button.

Using the location Object

The `location` object contains information about the current URL. It can also be used to enable navigation to different URLs on the Internet. The most frequently used property of the `location` object is the `HREF` property. The `HREF` property specifies the exact URL of the current document.

The properties of the `location` object consist of various pieces of information that make up a complete URL. The following syntax is used for specifying the complete URL:

```
protocol://hostname:port/pathname/
search#hash
```

The preceding syntax consists of the following attributes:

- `protocol`: Specifies the protocol, such as HTTP and FTP, which is used to transmit data over the Internet.
- `hostname`: Specifies the hostname of the URL, such as http://www.silvertech.com.
- `port`: Specifies the port number of the URL. It is optional. If the port number is not specified for the http request, the browser automatically connects to port 80.
- `pathname`: Specifies the path in the URL, such as http://www.silvertech.com/careers.html.
- `search`: Specifies the search string which is any portion of the URL followed by a question mark, such as http://www.silvertech.com/careers.html/search?programmer. The question mark is used for embedding arguments in the URL.
- `#hash`: Specifies the internal hyperlink of a Web page, such as http://www.silvertech.com/careers.html#position1.

The following table lists the most commonly used properties of the `location` object.

Properties	***Description***	***Syntax***
`hash`	*Specifies or returns the internal link anchor name. An internal link follows # in the URL.*	*`location.hash`*
`host`	*Specifies or returns the hostname: port portion of the URL.*	*`location.host`*

`hostname`	*Specifies or returns the hostname.*	`location.hostname`
`href`	*Specifies or returns the partial or full path of a file or a website.*	`location.href`
`port`	*Specifies or returns the port number.*	`location.port`
`protocol`	*Specifies or returns the protocol.*	`location.protocol`

The Properties of the location Object

The following table lists the most commonly used methods of the `location` object.

Methods	***Description***	***Syntax***
`assign()`	*Is used to load a new document.*	`location.assign(URL);`
`reload()`	*Is used to reload the current document.*	`location.reload();`
`replace()`	*Is used to replace the current document with another document.*	`location.replace (newURL);`

The Methods of the location Object

Consider the following code snippet to open the home page of XYZ.com website upon the click of a button:

```
<FORM>
<INPUT type = "button" VALUE = "visit
us" onClick = "location.href =
'http://www.xyz.com'">
</FORM>
```

In the preceding code snippet, the `HREF` property of the `location` object is set to the URL of the XYZ website. As a result, when the user clicks the button, the home page of the XYZ website is displayed.

Consider the following code snippet to retrieve information regarding the current URL:

```
document.write(location.protocol);
```

In the preceding code snippet, the protocol of the current URL is retrieved using the `protocol` property of the `location` object.

Working with Form Objects

Consider the scenario of BookYourHotel.com website. To purchase a product, a customer has to fill the purchase order form. The purchase order form requires entering the customer details, such as the name, gender, and address. This information is entered using the form objects, such as the text box and radio buttons. To access and process the content of the form objects, you need to refer to the form object.

The form object is a browser object, which acts as a container for several other objects that collect information from a user. The information collected using objects, such as the command buttons, radio buttons, text boxes, combo boxes, and check boxes, is submitted to the server for processing. Therefore, the form object enables you to create interactive Web pages.

The following objects of JavaScript are commonly used while working with Web forms:

- `form`
- `button`
- `checkbox`
- `text`
- `textarea`
- `radio`
- `select`

form

A form accepts user inputs. When a Web page containing multiple forms is viewed in a browser, the browser creates a `form` object for every form on the Web page. These forms can also be accessed using an array. For example, the Web browser can access the first form on the Web page using `document.forms[0]`, and the second form can be accessed as `document.forms[1]`.

The methods associated with the`form` object are:

- `submit()`: Submits a form to the server for processing. The following syntax is used for the `submit()` method:
  `form1.submit()`
  In the preceding syntax, `form1` is the name of the form that will be submitted to the server on the occurrence of an event, such as clicking a hyperlink.
- `reset()`: Resets all the fields of the given form. The following syntax is used for the `reset()` method:
  `form1.reset()`
  In the preceding syntax, `form1` is the name of the form. The `reset()` method clears all the information entered in various fields of the form.

The events associated with the `form` object are:

- `onsubmit`: Occurs when a user clicks a button or hyperlink to submit the form to the server for processing.
- `onreset`: Occurs when a user opts for resetting the fields of the form by clicking the reset button or hyperlink.

button

The `button` object refers to the HTML button. It allows you to perform several tasks based on user actions. You can write functions that are executed when a user clicks a specific button.

The methods associated with the button object are:

- click(): Simulates clicking of a button. For example,myButton.click().
- focus(): Sets focus on a button.
- blur(): Removes focus from the button.

The event attributes associated with the button object are:

- onclick: Occurs when a user clicks a button.
- onmouseDown: Occurs when a mouse button is pressed.
- onmouseUp: Occurs when the mouse button is released.

checkbox

The checkbox object refers to the HTML check box.

The methods associated with the checkbox object are:

- click()
- focus()
- blur()

The checkbox object can be referred inside a form using the following syntax:

```
document.form1.checkbox1
```

In the preceding syntax,form1 is the name of the form and checkbox is the name or ID of the check box.

Consider the following code snippet to refer the checkbox object inside a form and perform validation on it:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
body{
background-color:pink;
}
</STYLE>
<SCRIPT>
function validate()
{
if(document.f1.check1.checked==false)
{
alert("Please select the check box");
return false
}
else
return true
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="f1" method="get"
action="thanks.html" onsubmit="return
validate();" >
<INPUT type="checkbox" ID="check1" /
>Do you want to accept?
<INPUT type="submit" name="submit"
value="Submit"></INPUT>
<INPUT type="reset" name="reset"
value="Reset"></INPUT>
</FORM>
</BODY>
</HTML>
```

The preceding code snippet refers to the checkbox object inside a form and validates it before the user submits a form. It refers to the checkbox object using the document.f1.check1 syntax. It checks whether the user has selected the check box or not by using the document.f1.check1.checked==false condition. If the user has not selected the check box, the document.f1.check1.checked==false condition evaluates to false and the message, **Please select the check box** is displayed.

text

The text object refers to the HTML text box.

The methods associated with the text object are:

- blur(): Removes focus from a text field.
- focus(): Sets focus on a text field.
- select(): Selects a text field.

The event attributes associated with the text object are:

- onfocus: Fired when an element receives focus.
- onchange: Fired after the occurrence of the blur event, when the value of the text object is modified.
- Onselect: Fired when the user selects a part of the text within the text field.

The text object can be referred inside a form using the following syntax:

```
document.form1.text1
```

In the preceding syntax,form1 is the name of the form and text1 is the name or ID of the text box.

textarea

The textarea object refers to the HTML textarea.

The textarea object has similar methods and event attributes as that of the text object.

radio

The radio object refers to an HTML radio button.

The radio object can be referred inside a form using the following syntax:

```
document.form1.radio1
```

In the preceding syntax,form1, is the name of the form and radio1 is the name or ID of the radio button.

Consider the following code snippet to refer the radio object inside a form and perform validation on it:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
body{
background-color:pink;
}
</STYLE>
<SCRIPT>
function validate()
{
```

```
if((document.f1.M.checked==false)&&
(document.f1.F.checked==false))
{
alert("Please select the gender");
return false
}
else
return true
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="f1" method="get"
action="thanks.html" onsubmit="return
validate();" >
Gender:
<INPUT type="radio" name="gender"
ID="M" />Male
<INPUT type="radio" name="gender"
ID="F" />Female
<INPUT type="submit" name="submit"
value="Submit" align="center"></
INPUT>
<INPUT type="reset" name="reset"
value="Reset"></INPUT>
</FORM>
</BODY>
</HTML>
```

The preceding code snippet refers to the `radio` object inside a form and validates it before the user submits the form. It checks whether the radio buttons, **Male** or **Female**, for indicating gender have been selected or not. If the user has not indicated gender, the checked property of the radio button returns false and **Please select the gender** message is displayed.

select

The `select` object refers to the HTML drop-down list. The methods associated with the `select` object are:

- `blur()`
- `focus()`

The events attributes associated with the `select` object are:

- `onchange`
- `onfocus`
- `onblur`

The `select` object can be referred inside a form using the following syntax:

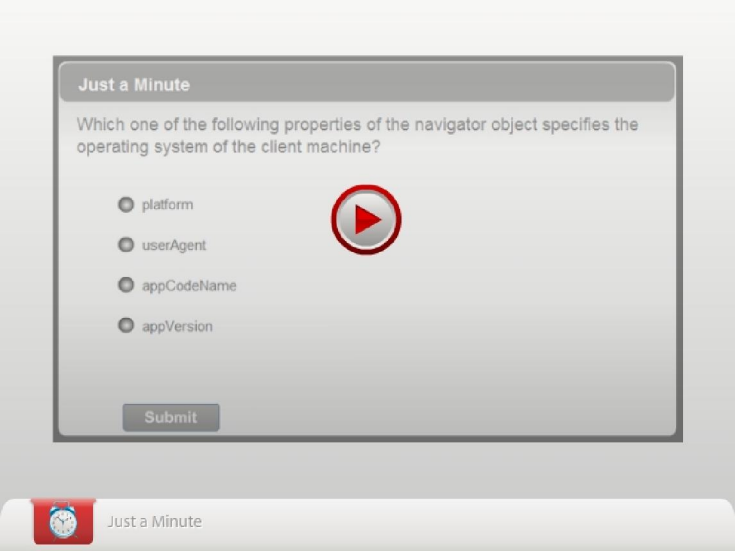
```
document.form1.select1
```

In the preceding syntax,`form1` is the name of the form and `select1` is the name or ID of the `select` object.

Consider the following code snippet to refer the `select` object inside a form and perform validation on it:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT>
function validate()
{
if(document.f1.opt.value==0)
{
alert("Please select the country");
return false
}
else
return true
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="f1" method="get"
action="thanks.html" onSubmit="return
validate();" >
Country:
<SELECT ID="opt" style="width:220">
<OPTION value=0>Select any country</
OPTION>
<OPTION value="1">India</OPTION>
<OPTION value="2">USA</OPTION>
<OPTION value="3">UK</OPTION>
</SELECT>
<INPUT type="submit" name="submit"
value="Submit">
<INPUT type="reset" name="reset"
value="Reset">
</FORM>
</BODY>
</HTML>
```

The preceding code snippet refers to the `select` object inside a form and validates it before the user submits a form. It refers to the `select` object using the `document.f1.opt statement`. It checks whether a user has indicated his/her country or not using the `document.f1.opt.value==0` condition. If the user has not indicated his/her country, the **Please select the country** message is displayed.



Activity 5.2: Manipulating the

Summary

In this chapter, you learned that:

- To create a form on a Web page, you need to use the `<FORM>` tag.
- The `<FORM>` tag supports the following attributes:
  - `name`
  - `ID`
  - `action`
  - `method`
  - `autocomplete`
  - `novalidate`
  - `target`
- The fields can be added to a form by using the following tags:
  - `<INPUT>`
  - `<SELECT>`
  - `<LABEL>`
  - `<FIELDSET>`
  - `<TEXTAREA>`
  - `<DATALIST>`
  - `<KEYGEN>`
  - `<OUTPUT>`
  - `<BUTTON>`
- JavaScript defines the following browser objects on a Web page:
  - `window`
  - `document`
  - `navigator`
  - `screen`
  - `history`
  - `location`
- Browser objects represent the browser environment and provide properties and methods for its access and manipulation.
- The form object is a browser object, which acts as a container for several other objects that collect information from a user.
- The following objects of JavaScript are commonly used while working with Web forms:
  - `form`
  - `button`
  - `checkbox`
  - `text`
  - `textarea`
  - `radio`
  - `select`

Reference Reading

Designing an HTML Form

Reference Reading: Books	*Reference Reading: URLs*
The Definitive Guide to HTML5 By Adam Freeman	*http://www.html5rocks.com/en/tutorials/forms/html5forms/* *http://www.ohio.edu/technology/training/upload/html-advanced-reference-guide.pdf*

Manipulating the Components of a Web Page

Reference Reading: Books	*Reference Reading: URLs*
The Definitive Guide to HTML5 By Adam Freeman	*http://www.w3schools.com/js/js_ex_browser.asp*

Chapter 6

Working with Graphics

You often need to add graphics, such as bitmap images or 2D shapes, on the pages of your website. In addition, you can add text and apply different colors and gradients to the graphics being added to the Web pages. The canvas element in HTML provides a drawing surface that allows you to add text, shapes, and images to the websites dynamically. In addition, you can add transition and animation effects to the graphics to make the Web page more visually appealing.

This chapter discusses a canvas and the various graphic objects that can be created on the canvas. In addition, it discusses how to apply animations and transformations on the graphic object of the canvas.

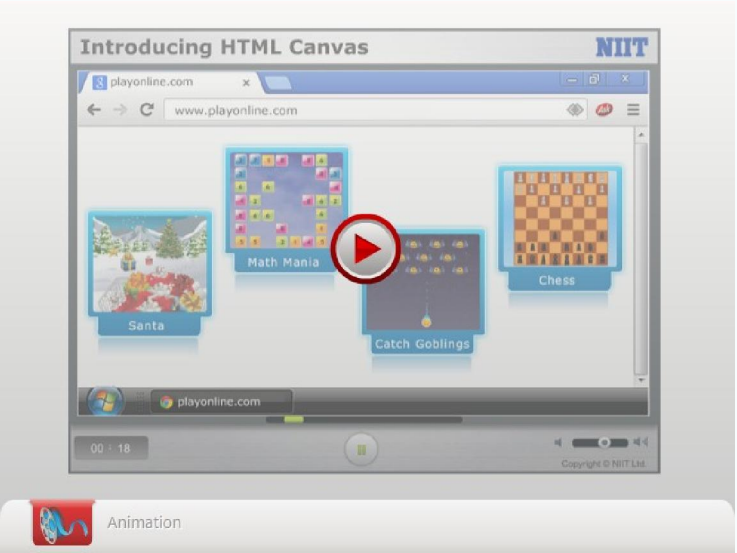
Objectives

In this lesson, you will learn to:

- Introduce canvas
- Transform and animate canvas elements

Introducing Canvas

LearnGraphs ltd. offers online tutorials on basic graphing skills. These tutorials include a wide variety of graphs, such as bar graph, histogram, and pie chart, which need to be drawn on their Web pages. However, to draw graphs on a Web page, you need to incorporate shapes, such as lines, arcs, circles, and apply effects, such as colors and gradients, on the shapes. With the introduction of canvas element in HTML, you can create such graphics and apply animations on a Web page easily. The canvas element has a huge set of functions in the form of a 2D Application Programming Interface (API) that are used to draw graphics on the canvas.



Creating a Canvas

Canvas provides an easy and powerful way to create graphics on a Web page. A canvas has no content of its own. A canvas is simply an area on a Web page that acts as a container for embedding graphic objects. It allows dynamic rendering of bitmap images and 2D shapes by using JavaScript. You need to perform the following tasks to create a canvas and use it for drawing a variety of graphics:

1. Define the Canvas
2. Access the Canvas

Defining the Canvas

A canvas is defined by using the `<CANVAS>` tag. This tag is defined in the body section of the HTML document. The `<CANVAS>` tag provides various attributes that enable you to specify the size, border, and ID for the canvas. The attributes provided by the `<CANVAS>` tag are listed in the following table.

Attribute	***Description***
`ID`	*Is used to specify a unique ID for the canvas that is used to identify the canvas in the JavaScript code.*
`width`	*Is used to specify the width for the canvas in pixels. The default value for the `width` attribute is 300 pixels.*
`height`	*Is used to specify the height for the canvas in pixels. The default value for the `height` attribute is 150 pixels.*
`style`	*Is used to define the style to be applied to the canvas.*

The Attributes of the `<CANVAS>` Tag

You can define a canvas by using the following code within the `<BODY>` tag:

```
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
```

The preceding code will create a canvas with ID, `myCanvas`, of size 300 x 300 with a solid black border of `1px` thickness.

Accessing the Canvas

Defining a canvas element only creates a blank drawing surface. However, to actually draw graphic objects on the canvas, you need to access the canvas in JavaScript code. You can write the following code in the `<BODY>` tag to access the canvas element:

```
<SCRIPT>
var c=document.getElementById
```

```
("myCanvas");
var ctx=c.getContext("2d");
</SCRIPT>
```

In the preceding code, the `getElementById()` method is used to return the reference of the element with ID,`myCanvas`, and stores the reference in the variable named c. Further, the `getContext()` method is used to return a drawing context object that provides the methods and properties needed for drawing graphics on the canvas. This object is stored inside the variable named `ctx`. The `getContext()` method accepts a string argument that specifies the type of canvas to be created. In the preceding code, `2d` is passed as an argument to the `getContext()` method as majority of the browsers support the creation of 2D graphics inside the canvas.

Working with Color, Shapes, and Styles

Consider a scenario of a website for Preschoolers, which provides basic learning solutions to kids. It provides simple online activities that assist kids to identify basic colors and shapes. For example, one of the activities displays a color picker and various shapes to the students. The students are asked to identify and fill a particular shape with the specified color as a part of this activity. To create such applications, you need to embed color, shapes, and styles on a Web page. Using canvas, this can be achieved by using the JavaScript predefined color and style properties and methods.

The introduction of Canvas has simplified the task of drawing objects, such as rectangles, on a Web page. You can easily draw these objects by using the JavaScript methods. Further, you can also specify the colors, gradients, or pattern styles to decorate the graphic objects on the canvas.

Working with Shapes

After creating a canvas, you can draw shapes, such as rectangle and square, on it. Rectangles and squares are the easiest shapes to draw on the canvas element by using the JavaScript functions. Using these functions, you can create a shape, clear a certain portion of the shape, and apply outline to the shape. For this, you can use the following methods:

- `rect()`
- `fillRect()`
- `strokeRect()`
- `clearRect()`

rect()

The `rect()` method is used to create a rectangle on the canvas. However, it picks the default color of the canvas to draw the outline of the rectangle. Therefore, the rectangle is not visible on the canvas. To make a rectangle visible on the canvas, you need to provide its outline or stroke color by using the `stroke()` method. This method uses the default black stroke to draw the rectangle.

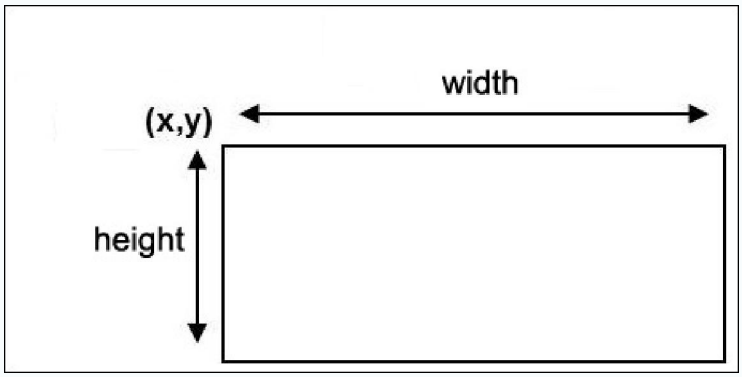
The syntax of using the `rect()` method to create a rectangle on a canvas is:

```
rect(x,y,width,height);
```

In the preceding syntax:

- `x`: Specifies the x-coordinate of the upper-left corner of the rectangle.
- `y`: Specifies the y-coordinate of the upper-left corner of the rectangle.
- `width`: Specifies the width of the rectangle, in pixels.
- `height`: Specifies the height of the rectangle, in pixels.

The following figure explains the dimensions of a rectangle.

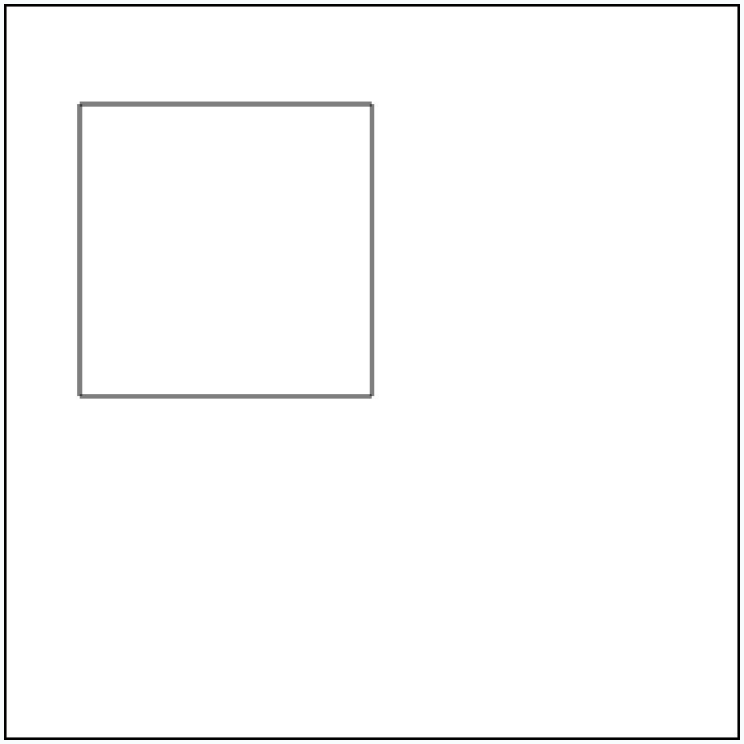


The Dimensions of a Rectangle

Consider the following code snippet for creating a rectangle on the canvas having ID, `myCanvas`:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.rect(30,40,120,120);
ctx.stroke();
</SCRIPT>
</BODY>
</HTML>
```

The preceding code snippet creates a rectangle of size 120 x 120 starting from the coordinates, (30, 40), on the canvas, as shown in the following figure.



The Rectangle Created on the Canvas

fillRect()

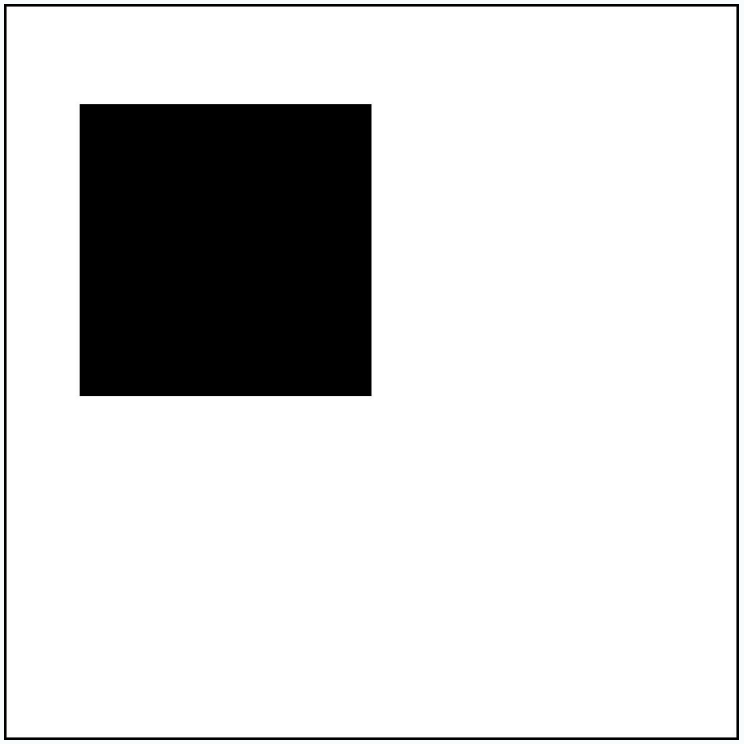
The `fillRect()` method is used to create a rectangle filled with the specified color. The default fill color is black. The following syntax is used to create a filled rectangle on a canvas:

```
fillRect(x,y,width,height);
```

Consider the following code for creating a filled rectangle on the canvas:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.fillRect(30,40,120,120);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code will create a rectangle filled with black color, as shown in the following figure.



A Rectangle Filled with Black Color

strokeRect()

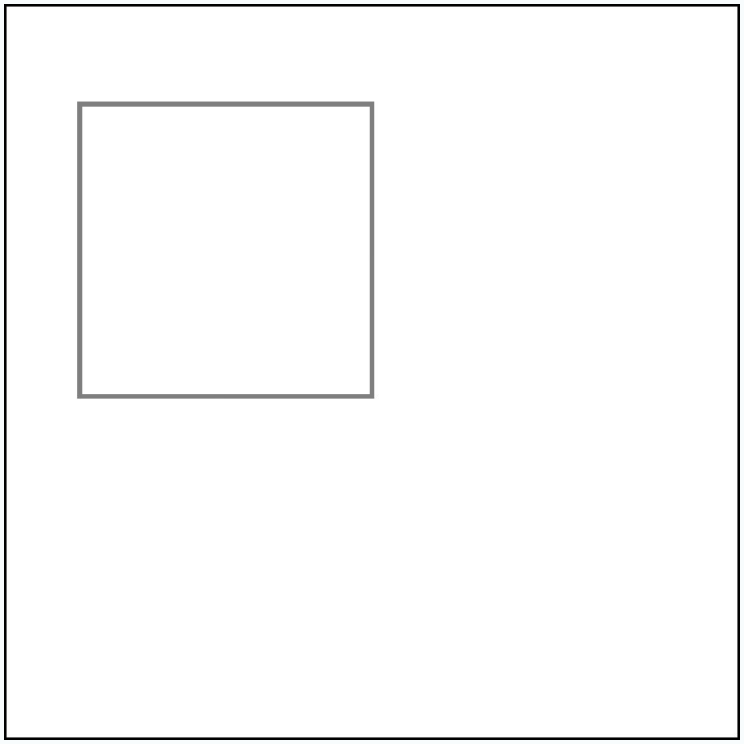
When you create a rectangle using by the `rect()` method, you also need to use the `stroke()` method to define its outline on the canvas. Instead of using two methods, `rect()` and `stroke()`, to draw a rectangle on the canvas, you can use a single method, `strokeRect()`, to draw a rectangle with the specified stroke color. By default, the `strokeRect()` method uses the black color to create an outline of the rectangle. The syntax of using the `strokeRect()` method to draw a rectangle on a canvas is:

```
strokeRect(x,y,width,height);
```

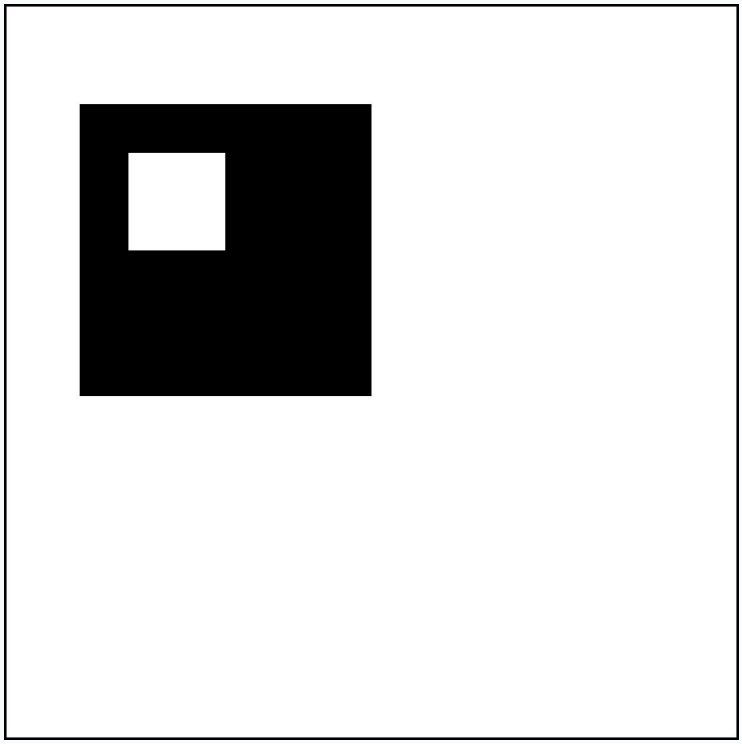
Consider the following code for drawing a rectangle on a canvas:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.strokeRect(30,40,120,120);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code will create a rectangle of size 120 x 120 at the position, (30, 40), on the canvas, as shown in the following figure.



The Rectangle Created by Using the strokeRect() Method



The Output Derived by Using the `clearRect()` Method

clearRect()

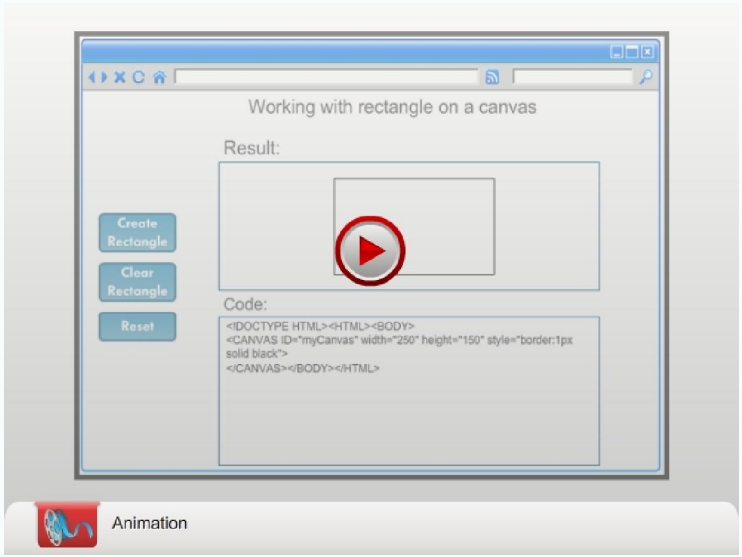
The `clearRect()` method is used to clear a portion of the rectangle. It clears the specified pixels within the given rectangle. The following syntax can be used to clear a rectangle on a canvas:

```
clearRect(x,y,width,height);
```

Consider the following code to clear a part of the rectangle created on the canvas having ID, `myCanvas`:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.fillRect(30,40,120,120);
ctx.clearRect(50,60,40,40);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code clears a portion of the rectangle of size 40 x 40 starting from the coordinates, (50, 60), as shown in the following figure.

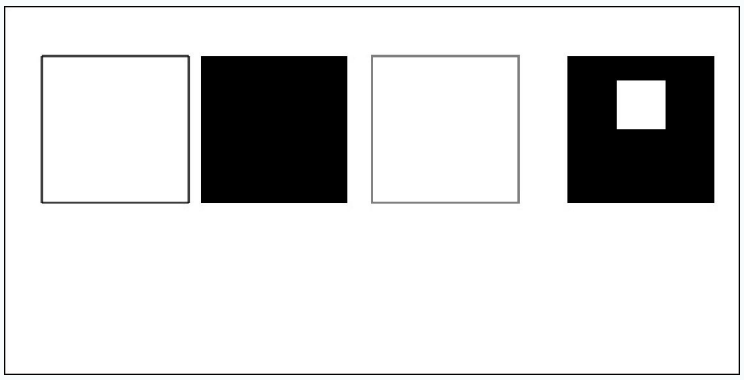


Now, consider the following code to draw rectangular shapes on the canvas using various methods:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="600"
height="300"
style="border:1px solid black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.rect(30,40,120,120);
ctx.stroke();
ctx.fillRect(160,40,120,120);
ctx.stroke();
ctx.strokeRect(300,40,120,120);
ctx.fillRect(460,40,120,120);
ctx.clearRect(500,60,40,40);
```

```
</SCRIPT>
</BODY>
</HTML>
```

The preceding code creates rectangles on the canvas, as shown in the following figure.



The Rectangles Created on the Canvas

Working with Colors

The graphic objects on a canvas are created by using the default stroke and fill color. However, you can use colors other than the default color for creating the graphic objects. The following properties can be used to apply colors on the canvas objects:

- `fillStyle`
- `strokeStyle`
- `shadowColor`

fillStyle

The `fillStyle` property is used to define a color that will be used to fill any closed shape drawn on the canvas. The default value of the `fillStyle` property is solid black. The following syntax is used to apply fill style on a graphic object:

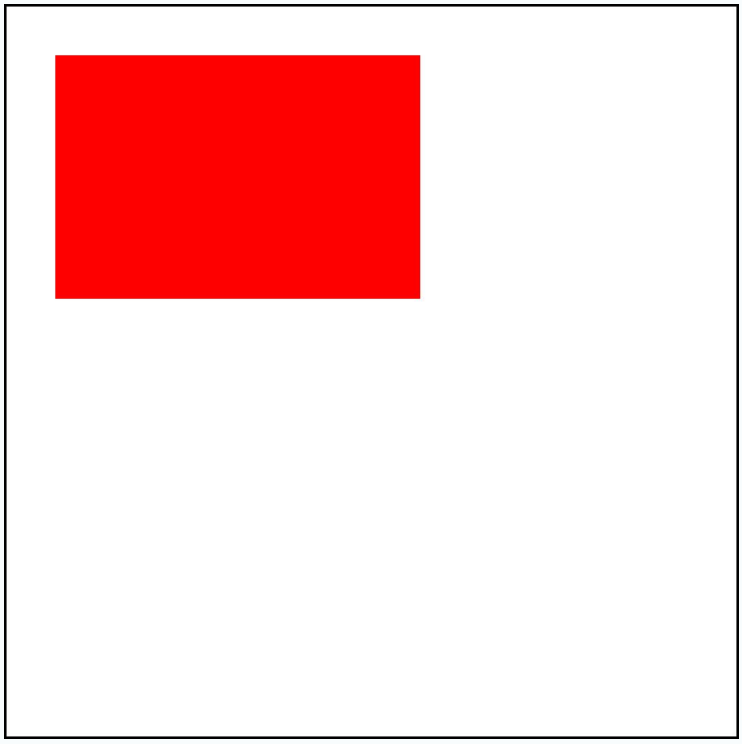
```
fillStyle="color";
```

In the preceding syntax, you can specify the color as red, green, or blue. In addition, you can also specify the hexadecimal value of the color ranging from 000000 to FFFFFF.

Consider the following code for applying fill style on a rectangle drawn on the canvas having ID, `myCanvas`:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(20,20,150,100);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code creates a rectangle of size 150 x 100 filled with red color at the position, (20, 20), on the canvas, as shown in the following figure.



A Rectangle Filled with the Red Color

strokeStyle

The `strokeStyle` property is used to set the outline color of a shape drawn on the canvas. The default value of the `strokeStyle` property is solid black. The following syntax can be used to apply stroke style on a graphic object:

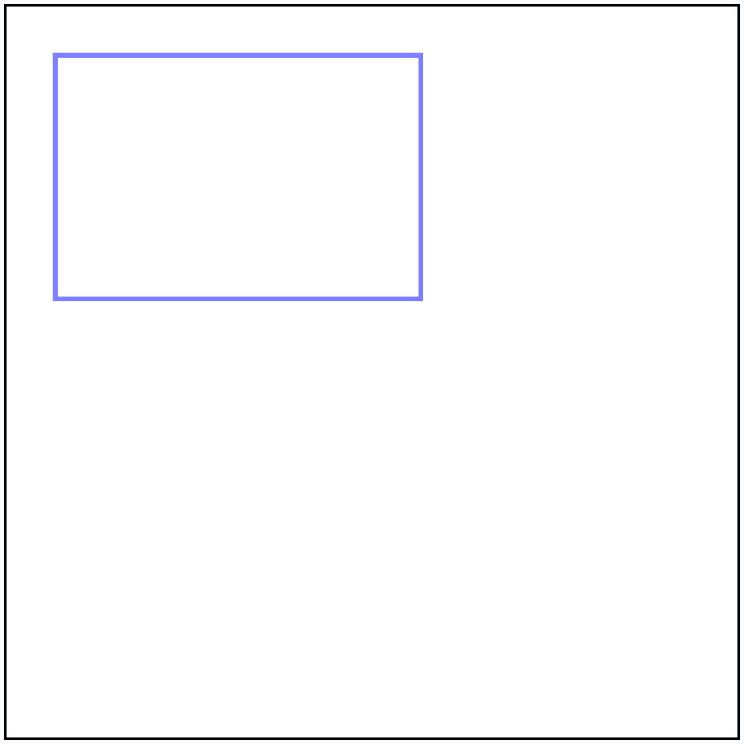
```
strokeStyle="color";
```

In the preceding syntax, `color` specifies the name or hexadecimal value of the color.

Consider the following code for applying the stroke style on a rectangle:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.strokeStyle="blue";
ctx.strokeRect(20,20,150,100);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code will create a rectangle of size 150 x 100 with its outline colored in blue at the position, (20, 20), on the canvas, as shown in the following figure.



The Rectangle with Strokes Colored in Blue

shadowColor

Once you have drawn a shape on the canvas, you may want to make it more stylish by casting a shadow on it. To cast a shadow of a graphic object on the canvas, you need to specify the color of the shadow. In addition, you need to specify how blurred you want your shadow to be. The `shadowColor` property is used to set the color for the shadows appearing on the graphic objects and the `shadowBlur` property is used to set the blur level for the shadows.

You can use the following syntax to use the `shadowColor` property:

```
shadowColor="color";
```

In the preceding syntax, `color` specifies the color that will be applied on shadows. The default value of the `shadowColor` property is solid black.

You can use the following syntax to define the shadowBlur property:

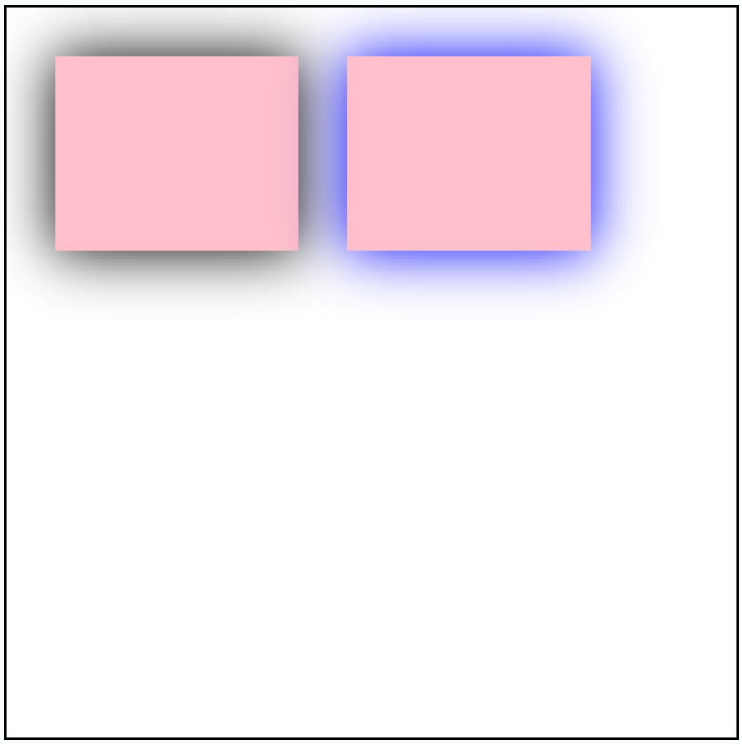
```
shadowBlur=number;
```

In the preceding syntax, `number` specifies the blur level of the shadow. It can accept the integer values, such as `1`, `2`, and `20`. Its default value is `0`.

Consider the following code for applying shadows on a rectangle:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.shadowBlur=40;
ctx.fillStyle="pink";
ctx.shadowColor="black";
ctx.fillRect(20,20,100,80);
ctx.shadowColor="blue";
ctx.fillRect(140,20,100,80);
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the blur level of the shadow of graphic objects is set to 40. In addition, the shadow color for the first rectangle is set to black and the shadow color for the second rectangle is set to blue. The output derived by using the `shadowBlur` and `shadowColor` properties is displayed in the following figure.



The Output Derived by Using the shadowBlur and shadowColor Properties

Working with Styles

Apart from creating simple shapes on the canvas, you can also apply styles, such as gradients, on them. A gradient is an object that provides smooth transition between two or more colors. To work with gradient styles, you can use the following methods:

- `addColorStop()`
- `createLinearGradient()`
- `createRadialGradient()`
- `createPattern()`

addColorStop()

To create gradients, you need to first specify the colors and their position in a gradient object. This is because the gradients are not visible until the colors are added to the objects. Therefore, to actually make the gradient effects visible on a graphic object, you need to add colors. You can add one or more colors on a gradient object by using the `addColorStop()` method. The `addColorStop()` method is used to specify the

colors and their corresponding positions in a gradient object. The following syntax can be used to define the `addColorStop()` method:

```
addColorStop(position,color);
```

In the preceding syntax:

- `position`: Specifies a value between 0.0 to 1.0 to represent the position from where to start and end the gradient color.
- `color`: Specifies the color that needs to be applied on the respective position.

The `addColorStop()` method is used along with the `createLinearGradient()` or `createRadialGradient()` method to display the gradients.

createLinearGradient()

The `createLinearGradient()` method is used to return a gradient object that represents a linear gradient for painting the specified color along a line. The following syntax can be used to apply a linear gradient:

```
createLinearGradient(x0,y0,x1,y1);
```

In the preceding syntax:

- `x0`: Specifies the x-coordinate of the start point of the gradient.
- `y0`: Specifies the y-coordinate of the start point of the gradient.
- `x1`: Specifies the x-coordinate of the end point of the gradient.
- `y1`: Specifies the y-coordinate of the end point of the gradient.

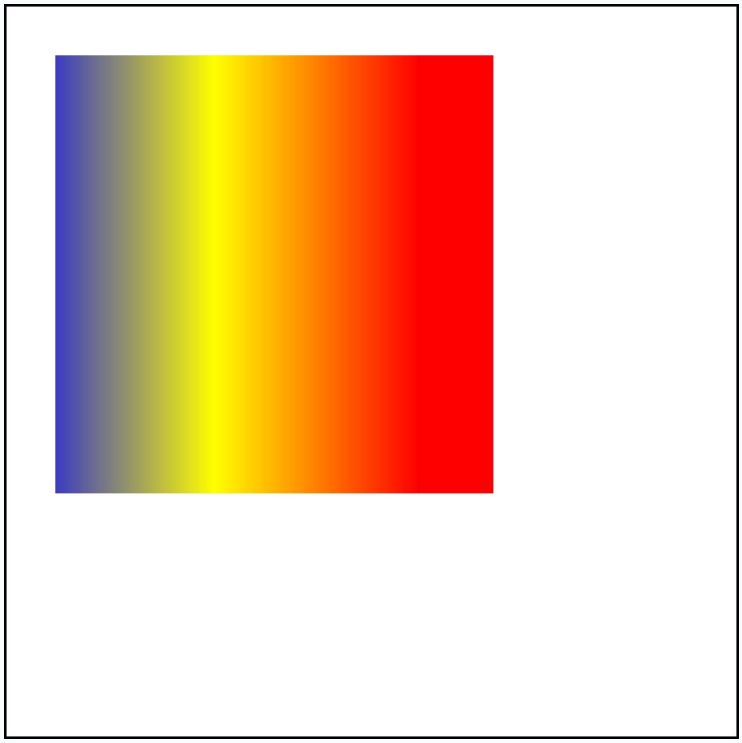
After creating the linear gradient object, you need to create the gradients by using the `addColorStop()` method. Once you have created the linear gradient, you need to apply it on a graphic object by using the following ways:

- Fill the graphic object with the linear gradient by using the `fillStyle` property.
- Apply the linear gradient on the outline of the graphic object by using the `strokeStyle` property.

Consider the following code for applying a linear gradient on a rectangle:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
var grad=ctx.createLinearGradient
(0,0,170,0);
grad.addColorStop(0,"blue");
grad.addColorStop("0.5","yellow");
grad.addColorStop(1,"red");
ctx.fillStyle=grad;
ctx.fillRect(20,20,180,180);
</SCRIPT>
</BODY>
</HTML>
```

In the preceding `code`, a gradient object is created by using the `createLinearGradient()` method. Further, the `addColorStop()` method is used to specify different colors to the gradient object, and then, the gradient object is passed to the `fillStyle` property to shade the rectangle in three different colors from left to right. The output derived by using the `createLinearGradient()` method is displayed, as shown in the following figure.



The Output Derived by Using the createLinearGradient() Method

createRadialGradient()

The `createRadialGradient()` method is used to return a gradient object that represents a radial or a circular gradient to be applied on a graphic object. A circular gradient paints colors along a cone specified by two circles, inner and outer. The following syntax can be used to apply a radial gradient:

```
createRadialGradient
(x0,y0,r0,x1,y1,r1);
```

In the preceding syntax:

- `x0`: Specifies the x-coordinate of the start point of the gradient.
- `y0`: Specifies the y-coordinate of the start point of the gradient. (x0,y0) specifies the center coordinate of the first circle of the cone.
- `r0`: Specifies the radius of the starting circle.
- `x1`: Specifies the x-coordinate of the end point of the gradient.
- `y1`: Specifies the y-coordinate of the end point of the gradient. (x1,y1) specifies the center coordinate of the second circle of the cone.

- `r1`: Specifies the radius of the ending circle.

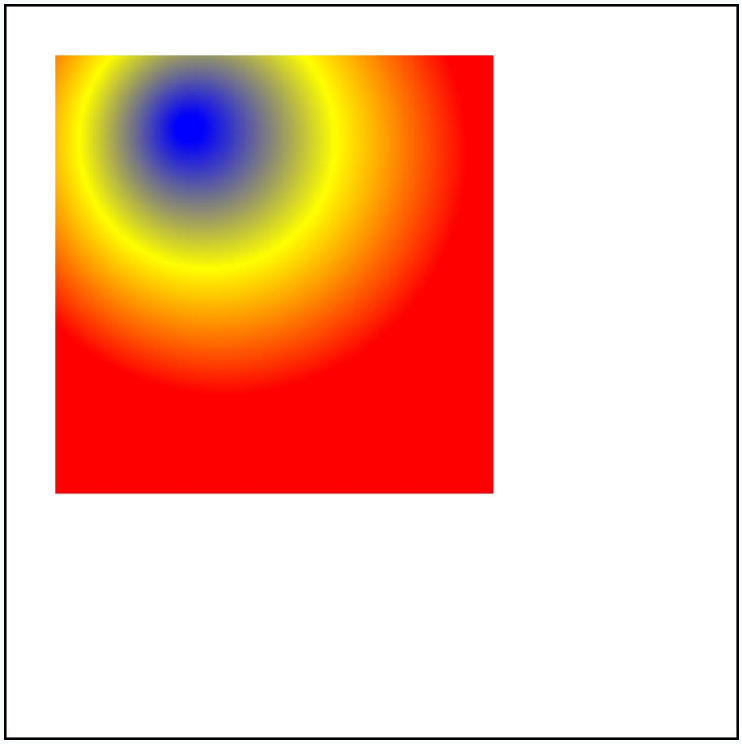
After creating the radial gradient object, you need to create the gradients by using the `addColorStop()` method. Once you have created the radial gradient, you need to apply it on a graphic object by using the following ways:

- Fill the graphic object with the radial gradient by using the `fillStyle` property.
- Apply the radial gradient on the outline of the graphic object by using the `strokeStyle` property.

Consider the following code snippet for applying a radial gradient on a rectangle:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300"style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
var grad=ctx.createRadialGradient
(75,50,5,90,60,100);
grad.addColorStop(0,"blue");
grad.addColorStop("0.5","yellow");
grad.addColorStop(1,"red");
ctx.fillStyle=grad;
ctx.fillRect(20,20,180,180);
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, a gradient object is created using the `createRadialGradient()` method. Further, the `addColorStop()` method is used to specify different colors to the gradient object and then, the gradient object is passed to the `fillStyle` property to shade the rectangle in three different colors along the radius given for the circle. The output derived by using the `createRadialGradient()` method is displayed in the following figure.



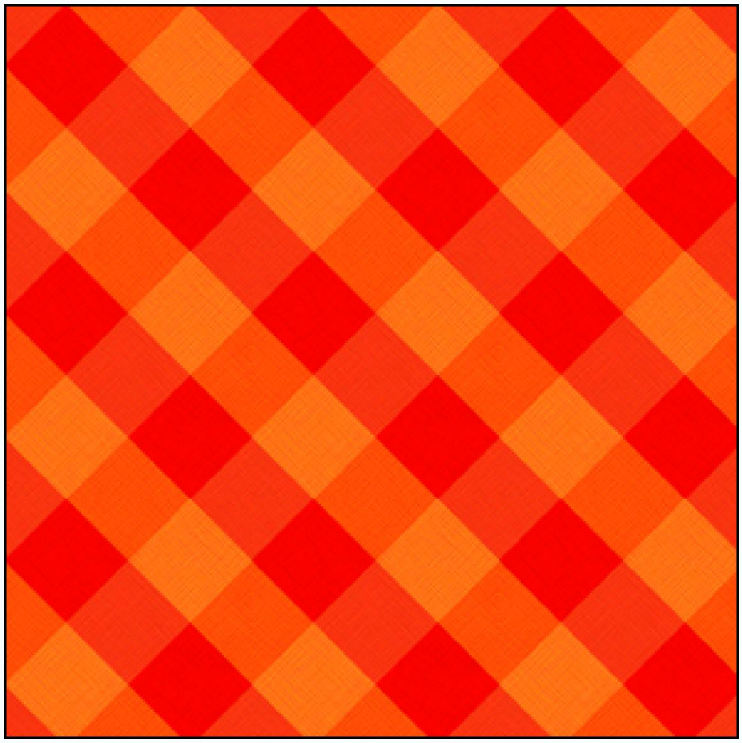
The Output Derived by Using the createRadialGradient() Method

createPattern()

The `createPattern()` method is used to create a pattern by displaying an image repeatedly on a canvas in the specified direction. For example, consider the following image.

The Image

If the preceding image is repeated vertically and horizontally, you can create a pattern, as shown in the following figure.



A Pattern

The following syntax can be used to create a pattern:

```
createPattern(img, "repeat|repeat-x|
repeat-y|no-repeat");
```

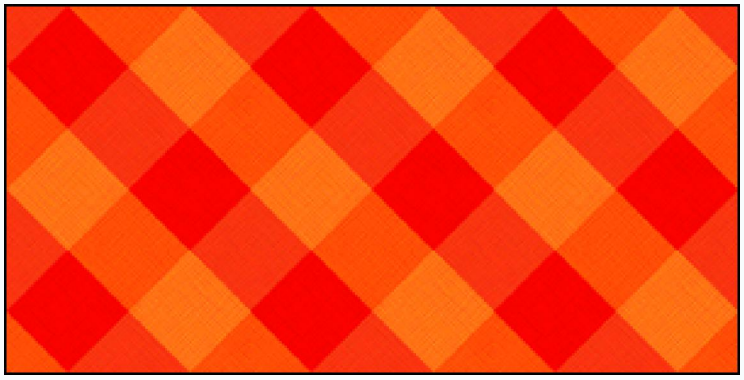
In the preceding syntax:

- `img`: Specifies the image or video to be used to create a pattern.
- `repeat`: Specifies that the pattern should be repeated horizontally and vertically.
- `repeat-x`: Specifies that the pattern should be repeated horizontally.
- `repeat-y`: Specifies that the pattern should be repeated vertically.
- `no-repeat`: Specifies that the pattern should be displayed only once.

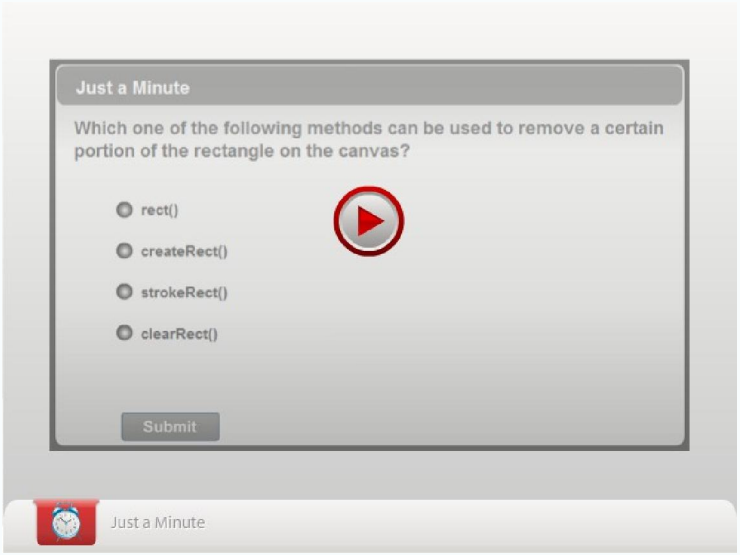
Consider the following code snippet for repeating an image horizontally and vertically:

```
<P>Image to use:</P>
<IMG src="pattern.png" ID="pattern">
<P>Canvas:</P>
<BUTTON onclick="draw
('repeat')">Repeat</BUTTON>
<CANVAS ID="myCanvas" width="300"
height="150" style="border:1px solid
#d3d3d3;">
Your browser does not support the
HTML5 canvas tag.</CANVAS>
<SCRIPT>
function draw(direction)
{
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
var img=document.getElementById
("pattern")
var pat=ctx.createPattern
(img,direction);
ctx.rect(0,0,300,150);
ctx.fillStyle=pat;
ctx.fill();
}
</SCRIPT>
```

The preceding code snippet repeats the `pattern.png` image horizontally and vertically in the rectangular area on the canvas, when the user clicks the **Repeat** button, as shown in the following figure.



Repeating Image on the Canvas



Working with Path, Text, and Images

LearnGraphs Ltd offers tutorials on creating pie charts. For this, a pie chart along with its caption, needs to be drawn. However, a pie chart is a circular chart that is divided into various sectors. To create such a pie chart, you need to draw a circle. In addition, to divide the circle into various sectors, you need to draw lines. In canvas, you can create such shapes, such as circles, lines, arcs, and triangles, by using path methods and properties.

Further, to insert the caption for the pie chart, you need to insert text. You can insert text in canvas by using the text properties and methods. In addition, you can also add images to the canvas.

Working with Path

A path is a series of points joined together to create lines or shapes. In canvas, you can use lines or paths to draw shapes other than rectangles or squares. Using paths or lines, you can create shapes such as circles, polygon, or triangles. However, to create a path, you need to first start or begin the path. Next, you need to call methods, such as `moveTo()` and `lineTo()`, to actually draw the path. Finally, you need to end or close the path so that the shape gets created. To create shapes by using paths, you can use various methods. These methods are described in the following table.

Method	***Description***
`fill()`	*Is used to fill the path on the canvas. The default color is black. You can change the fill color by using the* `fillStyle` *property.*
`stroke()`	*Is used to draw the outlines of the path. The default color is black. You can change the outline color by using the* `strokeStyle` *property.*

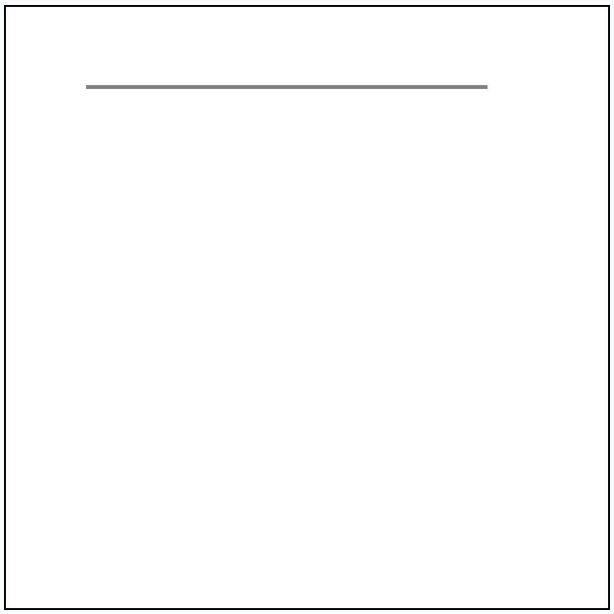
beginPath()	*Is used to begin a path or resets the current path.*
moveTo(x,y)	*Is used to move the path to the (x,y) coordinates on the canvas.*
closePath()	*Is used to create a path from the current position back to the starting position.*
lineTo(x,y)	*Is used to create a line from the starting position to the ending position specified by (x,y) coordinates. The starting position is defined by the (x,y) coordinates specified in the moveTo() method.*
clip()	*Is used to clip a specified area from the canvas.*
arc(x,y,r,sAngle, eAngle,counterclockwise)	*Is used to create an arc or a curve on the coordinate points (x,y) with the radius, r from the starting angle, sAngle to the end angle, eAngle on the canvas. The last parameter specifies whether the drawing should be counterclockwise or clockwise. The false value specifies clockwise and true value specifies counter-clockwise.*
arcTo (x1,y1,x2,y2,r)	*Is used to create an arc between two coordinate points, (x1,y1) and (x2,y2) with radius r on the canvas.*

The Path Methods

Consider the following code for creating a line on a canvas:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.moveTo(40, 40);
ctx.lineTo(240, 40);
ctx.stroke(); </SCRIPT>
</BODY>
</HTML>
```

In the preceding code , the `beginPath()` method will begin the path on the canvas. Further, the `lineTo()` method will draw a straight line from the canvas coordinates, `(40,40)`, specified in the `moveTo()` method upto the coordinates, `(240,40)`, as shown in the following figure.

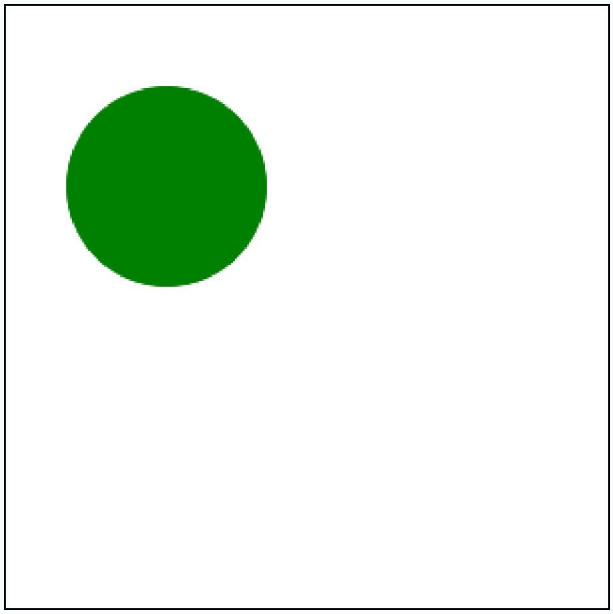


The Line Drawn on a Canvas

Consider the following code for creating a circle on a canvas:

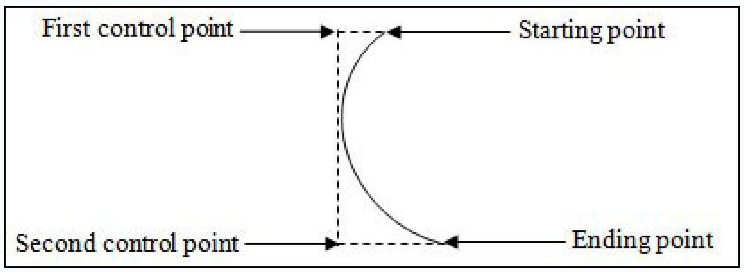
```
<!DOCTYPE HTML>
<HTML> <BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.arc(80, 90, 50, 0, Math.PI*2,
false);
ctx.closePath();
ctx.fillStyle="green";
ctx.fill();
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, a circle of the radius, 50, filled with the green color is drawn on the canvas at the coordinates, (80, 90), as shown in the following figure.



The Output Derived by Using the `arc()` Method

In addition to drawing lines, arcs, or circles, you can also draw more complex curvatures, such as the bezier curves. A bezier curve is defined with a context or starting point, two control points, and an ending point, as shown in the following figure.



The Bezier Curve

You can create the bezier curves by using the path method, `bezierCurveTo()`. The following syntax can be used to create a bezier curve:

```
bezierCurveTo
(cp1x,cp1y,cp2x,cp2y,x,y);
```

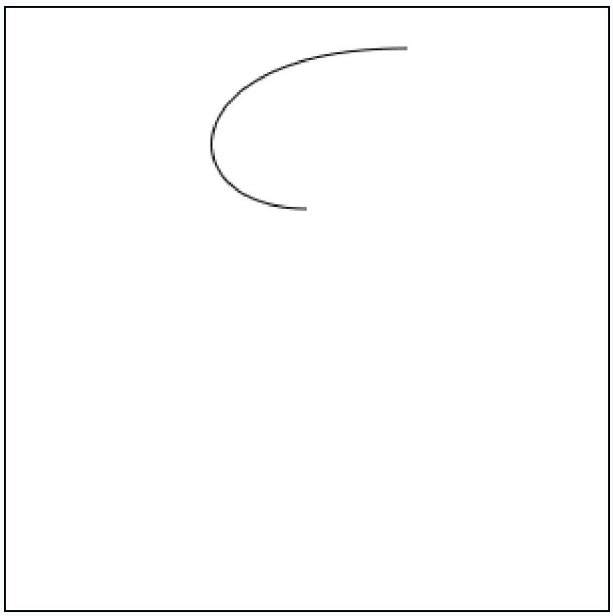
In the preceding syntax:

- `cp1x`: Specifies the x-coordinate of the first control point.
- `cp1y`: Specifies the y-coordinate of the first control point.
- `cp2x`: Specifies the x-coordinate of the second Bezier control point.
- `cp2y`: Specifies the y-coordinate of the second Bezier control point.
- `x`: Specifies the x-coordinate of the ending point.
- `y`: Specifies the y-coordinate of the ending point.

Consider the following code:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="300"
height="300"
style="border:1px solid black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
('myCanvas');
var ctx=c.getContext('2d');
ctx.beginPath();
ctx.moveTo(200,20);
ctx.bezierCurveTo
(80,20,80,100,150,100);
ctx.stroke();
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, a bezier curve is drawn starting from the point, (200, 20), to the point, (150, 100), as shown in the following figure.



The Bezier Curve

Working with Text

In addition to drawing shapes or lines on the canvas, you can also draw text on it. You can also apply different styles on the text. For this, you can use the various text properties. The following table lists the text properties.

Property	***Value***	***Description***
`font`	`font-style|font-variant|font-weight|font-size|font-family`	*Is used to set the font for the text on the canvas.*
`textAlign`	`start|end|center|left|right`	*Is used to set the alignments for the text.*
`textBaseLine`	`top|hanging|middle|bottom|alphabetic`	*Is used to set the baseline for the text where it will be drawn relative to its starting point.*

The Text Properties

The preceding properties can be used to decorate the text. However, you need to use the following methods to actually draw a text on a canvas:

- `fillText()`
- `strokeText()`

fillText()

The `fillText()` method is used to draw a text filled with solid color on a canvas. The default value for `fillText()` is black. The following syntax can be used to draw a filled text:

```
fillText(text,x,y,width);
```

In the preceding syntax:

- `text`: Specifies the text to be written on the canvas.
- `x`: Specifies the x-coordinate of the starting point of the text.
- `y`: Specifies the y-coordinate of the starting point of the text.
- `width`: Specifies the width of the text.

Consider the following code for drawing a filled text on the canvas:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas" width="200"
height="200" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.font="15px Georgia";
ctx.fillText("Fill Text",10,60,120);
ctx.font="15x Verdana";
var gradient=ctx.createLinearGradient
(0,0,170,0);
gradient.addColorStop("0","magenta");
gradient.addColorStop("0.5","blue");
gradient.addColorStop("1.0","red");
ctx.fillStyle=gradient;
ctx.fillText("Fill Text with
Gradient",10,90,120);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code sets the font style of the text to `15 px Georgia` by using the `font` property and draws the text, `Fill Text`, at the canvas coordinates, `(10,60)`, on the canvas. Similarly, it draws the text, `Fill Text with Gradient`, at the canvas coordinates, `(10, 90)`, and applies the gradient on it by using the `createLinearGradient()` method and the `fillStyle` property, as shown in the following figure.

The Output Derived by Using the fillText() Property

strokeText()

The `strokeText()` method is used to draw a text at the specified position on the canvas by using the current font style and color. The default outline color used by this method to draw a text on the canvas is black. The syntax to use the `strokeText()` method is:

```
strokeText(text,x,y,width);
```

In the preceding syntax:

- `text`: Specifies the text to be written on the canvas.
- `x`: Specifies the x-coordinate of the starting point of the text.
- `y`: Specifies the y-coordinate of the starting point of text.
- `width`: Specifies the width of the text.

Consider the following code:

```
<!DOCTYPE HTML>
<HTML>
  <BODY>
<CANVAS ID="myCanvas" width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.font="25px Georgia";
ctx.strokeText("Stroke
Text",10,60,120);
ctx.font="25x Verdana";
var gradient=ctx.createLinearGradient
(0,0,170,0);
gradient.addColorStop("0","magenta");
gradient.addColorStop("0.5","blue");
gradient.addColorStop("1.0","red");
ctx.strokeStyle=gradient;
ctx.strokeText("Stroke Text with
Gradient",10,90,200);
</SCRIPT>
</BODY>
</HTML>
```

The preceding code sets the font style of the text to `25px Georgia` by using the font property and draws the text, `Stroke text`, at the canvas coordinates,

(10,60), on the canvas. Similarly, it draws another text, Stroke Text with Gradient, at the canvas coordinates, (10,90), and applies the gradient on it by using the createLinearGradient() method and the strokeStyle property, as shown in the following figure.

The Output Derived by Using the strokeText() Method

Working with Images

In canvas, you can also draw images, image clips, or render videos. For this, you can use the drawImage() method. The drawImage() method is used to draw an image or a video on the canvas. In addition, it enables you to draw a part of an image on the canvas. To define the drawImage() method, you can use the following syntaxes:

```
drawImage(img,x,y);
drawImage(img,x,y,width,height);
drawImage
(img,sx,sy,swidth,sheight,x,y,width,h
eight);
```

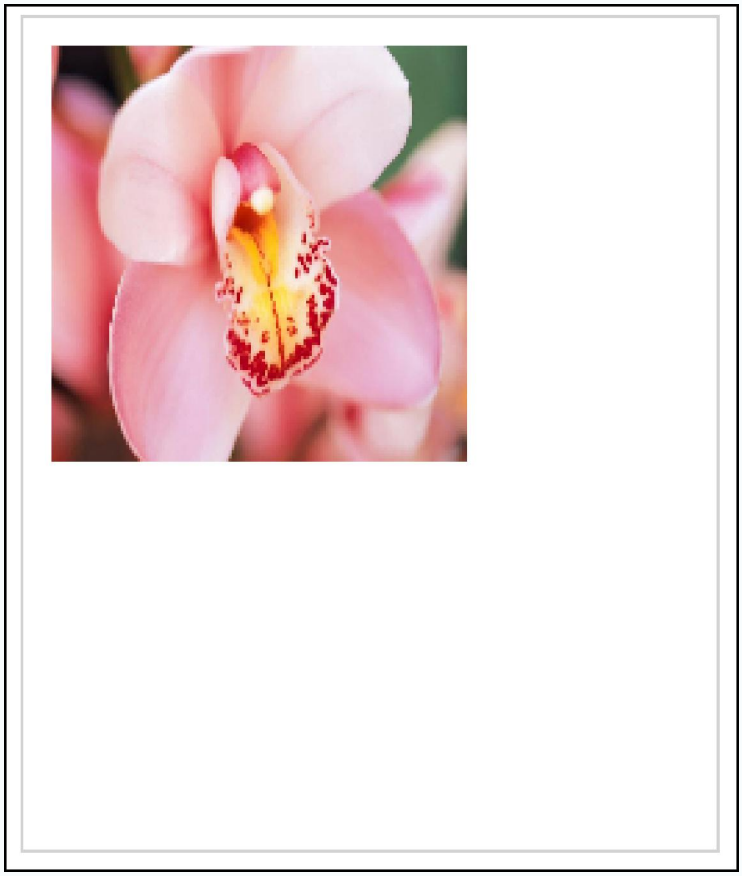
In the preceding syntax:

- q img: Specifies an image that needs to be drawn on the canvas.
- q x: Specifies the x-coordinate for the starting point of the image.
- q y: Specifies the y-coordinate for the starting point of the image.
- q width: Specifies the width of the image. It is an optional argument. If not specified, the actual width of the image is taken by default.
- q height: Specifies the height of the image. If not specified, the actual height of the image is taken by default.
- q sx: Specifies the x-coordinate where to start the clipping of the image.
- q sy: Specifies the y-coordinate where to start the clipping of the image.
- q swidth: Specifies the width of the clipped image.
- q sheight: Specifies the height of the clipped image.

Consider the following code to draw an image on a canvas:

```
<!DOCTYPE HTML>
<HTML>
<BODY onload=setImage()>
<CANVAS ID="myCanvas" width="250"
height="300"
style="border:1px solid #d3d3d3;">
Your browser does not support the
HTML5 canvas tag.</CANVAS>
<SCRIPT>
function setImage(){
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
var img=new Image();
img.src="orchid.jpg";
img.onload = function() {
  ctx.drawImage(img,10,10,150,150);
}
}
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the image is drawn on the canvas at the point, (10, 10), and of size 150 x 150, as shown in the following figure.



Drawing an Image on the Canvas

Working with Graphs

A graph is a way of representing relationships among a collection of items. A graph consists of a set of objects, called nodes, which are connected by links called edges. In a canvas, you can create graphs, such as bar graph or pie chart, to represent relationships. These graphs can be created by using methods provided to draw the shapes. However, to draw graphs by using these methods is a

tedious task. To simplify this task, you can use the various freely-downloadable JavaScript libraries. One such library is RGraph. To create graphs by using the RGraph library, you need to download this light-weight JavaScript library and save it in your system. RGraph allows you to create different types of graphs on a Web page.

Once the JavaScript library is downloaded, it can be referred to in a Web page by using the `<SCRIPT>` tag in the head section of the Web document. The following syntax is used to specify the jQuery library:

```
<SCRIPT type="text/javascript"
src="<RGraph_file_name>" ></SCRIPT>
```

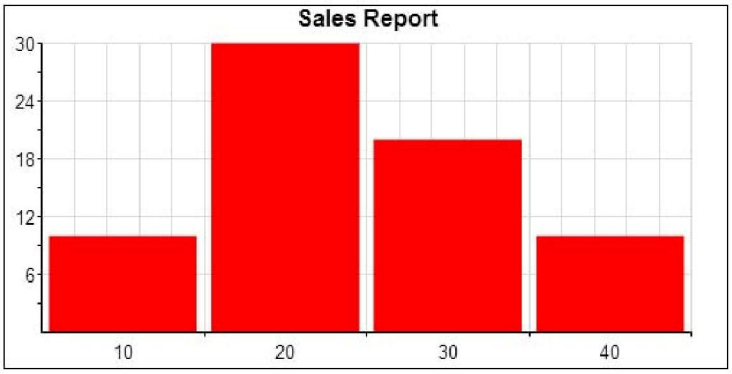
In the preceding syntax:

- The `<SCRIPT>` tag instructs the browser that the HTML document uses a script.
- The `type` attribute specifies the type of scripting used.
- The `src` attribute specifies the name of the JavaScript library used. If the JavaScript library is stored at the same location as the HTML document, only the name of the library can be given. However, if the JavaScript library and the HTML document are stored at different locations, then you have to specify the complete path of the JavaScript library.

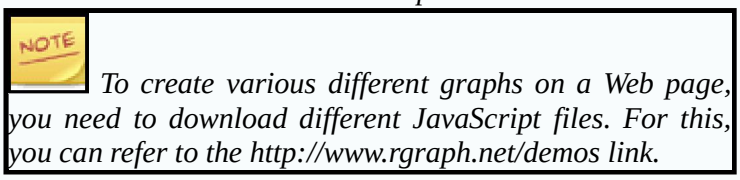
For example, you want to create a bar graph on a Web page. For this, you need to download the **RGraph.common.core.js** and **RGraph.bar.js** files. Consider the following code to create a bar graph on a Web page:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript"
src="RGraph.common.core.js" ></
SCRIPT>
<SCRIPT type="text/javascript"
src="RGraph.bar.js" ></SCRIPT>
  <TITLE>Bar chart</TITLE>
</HEAD>
<BODY>
<CANVAS width="500" height="250"
ID="test" style="border:1px solid
black"></CANVAS>
<SCRIPT type="text/javascript"
charset="utf-8">
var bar = new RGraph.Bar('test',
[10,30,20,10]);
bar.Set('chart.colors', ['red']);
bar.Set('chart.title', "Sales
Report" );
bar.Set('chart.labels', ["10" ,
"20" , "30" , "40" ]);
bar.Draw();
</SCRIPT>
</BODY>
</HTML>
```

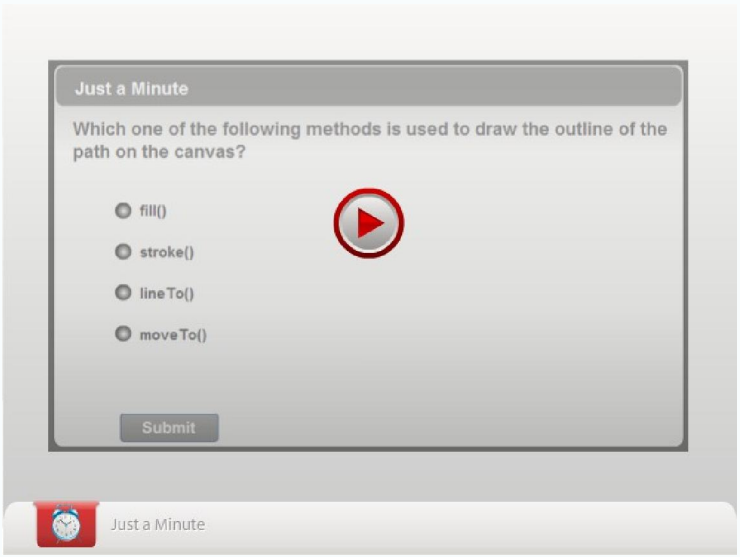
The preceding code creates a graph on the canvas, as shown in the following figure.



A Bar Graph



To create various different graphs on a Web page, you need to download different JavaScript files. For this, you can refer to the http://www.rgraph.net/demos link.



Activity 6.1: Introducing Canvas

Transforming and Animating Canvas Elements

LearnGraphs Ltd. wants to make the graphics eye-catching visually appealing to the users. For this, they need to animate or transform the shapes, such as scaling or rotating the line drawn on the canvas. To apply such animations, you need to apply animation effects on the shapes. With canvas, you can transform, rotate, or scale graphic objects.

Transforming Canvas Elements

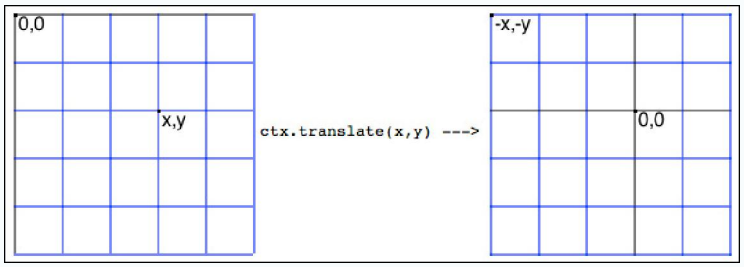
With CSS, you can move HTML elements from one position to another easily. However, while working with

a canvas, you may want to shift the graphics objects from one position to another and increase or decrease their size. This can be done by transforming the canvas elements. To transform the canvas elements, you can use the following methods:

- `translate()`
- `scale()`
- `rotate()`

Translate

The `translate()` method is used to reset the origin of the canvas to the specified position, as shown in the following figure.



The Resetting the Origin of the Canvas

In the preceding figure, the origin of the canvas is shifted to the (x,y) coordinate. The `translate()` method enables you to move all the graphic objects on the canvas by using just one method. For example, you have drawn a complex drawing onto the canvas and you need to move the drawing around on the canvas. To perform such a task, you need to adjust the x and y positions of all the graphic objects involved in the drawing. It is a time-consuming and error-prone task. You can simplify this task by just translating the context or the origin of the canvas to the new position and then, using the original x and y positions of all the graphic objects, redraw them on the canvas. This way, the graphic objects will be redrawn by taking into account the position of the new origin. Therefore, the graphic objects will be moved to the desired location on the canvas by using just one method call.

The syntax of the `translate()` method is:

```
translate(x,y);
```

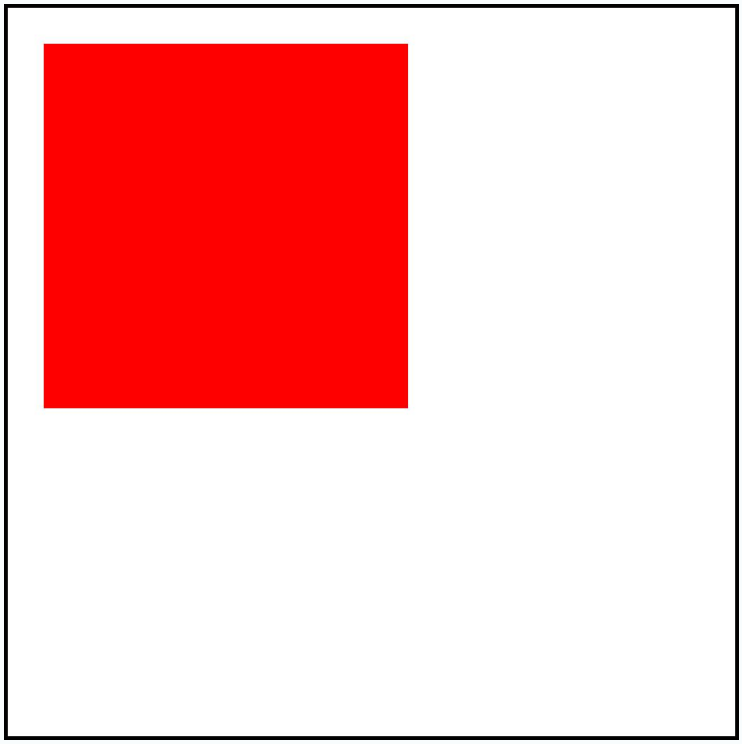
In the preceding syntax:

- `x`: Specifies the value to be added to the existing value of the x coordinate.
- `y`: Specifies the value to be added to the existing value of the y coordinate.

For example, you have created a rectangle by using the following code snippet:

```
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(10,10,100,100);
```

The preceding code snippet will draw a rectangle at the coordinate, (10, 10), as shown in the following figure.

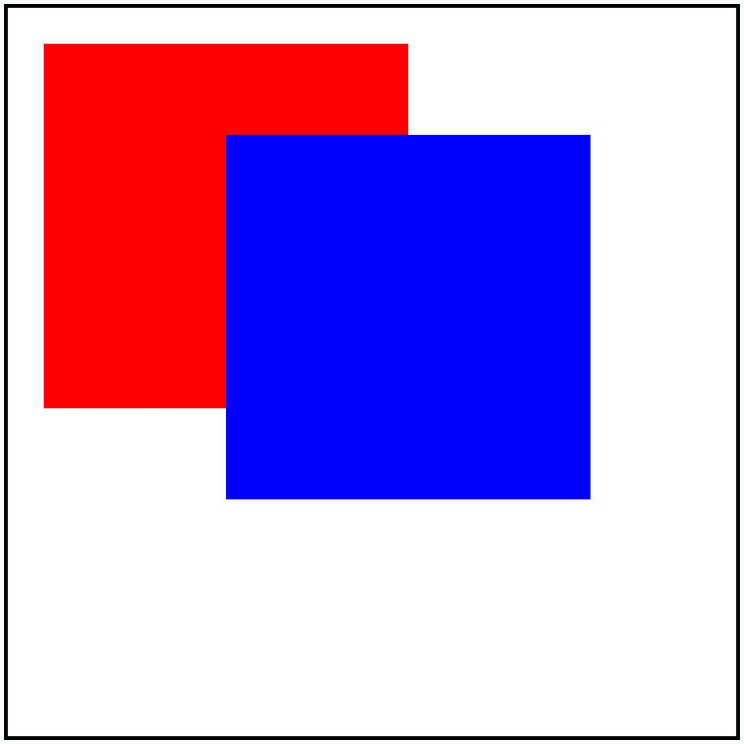


The Rectangle

Now, you want that the rectangle should be redrawn starting from coordinate, (60, 35). For this, you can use the `translate()` method, as shown in the following code snippet:

```
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(10,10,100,100);
ctx.translate(50,25);
ctx.fillStyle="blue";
ctx.fillRect(10,10,100,100);
```

In the preceding code snippet, the first rectangle is drawn in the red color at the point, (10, 10) from the origin of the canvas. The `translate()` method then resets the origin of the canvas at the point, (50, 25). The second rectangle is drawn in blue color at the point, (10, 10) from the new origin, (50, 25). Therefore, the second rectangle will be drawn at the point, (60, 35) from the point, (0, 0) of the canvas, as shown in the following figure.



The Output Derived by Using the translate() Method

Scale

The `scale()` method is used to increase or decrease the units in the canvas grid. This will allow you to draw scaled down or enlarged graphic objects. The following syntax can be used to scale the graphic objects:

```
scale(scalewidth,scaleheight);
```

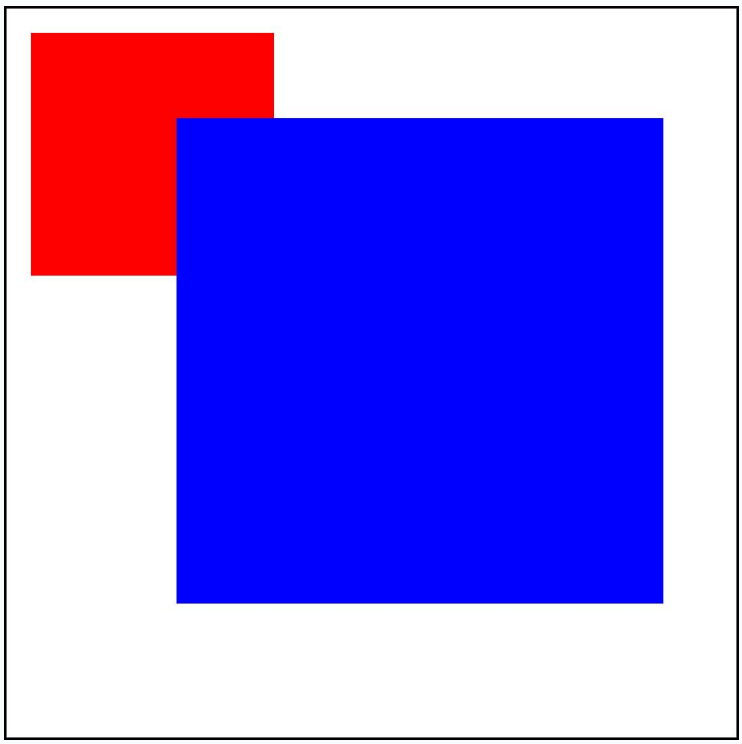
In the preceding syntax:

- `scalewidth`: Specifies the width (in percentage), a graphic object should be scaled to.
- `scaleheight`: Specifies the length (in percentage), a graphic object should be scaled to.

Consider the following code for scaling a rectangle on canvas:

```
<!DOCTYPE HTML>
<HTML>
  <BODY>
<CANVAS ID="myCanvas"  width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(10,10,100,100);
ctx.translate(50,25);
ctx.scale(2,2);
ctx.fillStyle="blue";
ctx.fillRect(10,10,100,100);
</SCRIPT>
</BODY>
</HTML>
```

In the preceding code, the red colored rectangle of size 100 x 100 is created starting from the canvas coordinates, (10, 10). The origin of the canvas is reset at the point, (50, 25), by using the `translate()` method. Then, the width and height of the rectangle are scaled by the factor of 2 by using the `scale()` method. Therefore, the width and height of the blue colored rectangle gets doubled and is redrawn from the canvas coordinates, (60, 35), as shown in the following figure.



The Output Derived by Using the scale() Method

Rotate

The `rotate()` method is used to rotate the graphic object to a specified degree in the clockwise direction. The following syntax can be used to apply rotation:

```
rotate(angle);
```

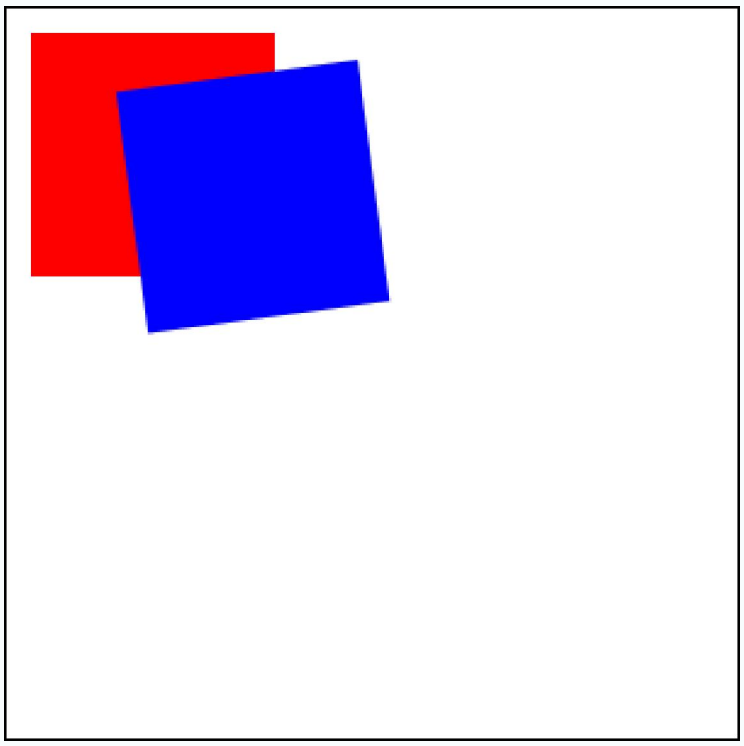
In the preceding syntax, `angle` specifies the degree a graphic object should be rotated to.

Consider the following code for applying rotations on canvas:

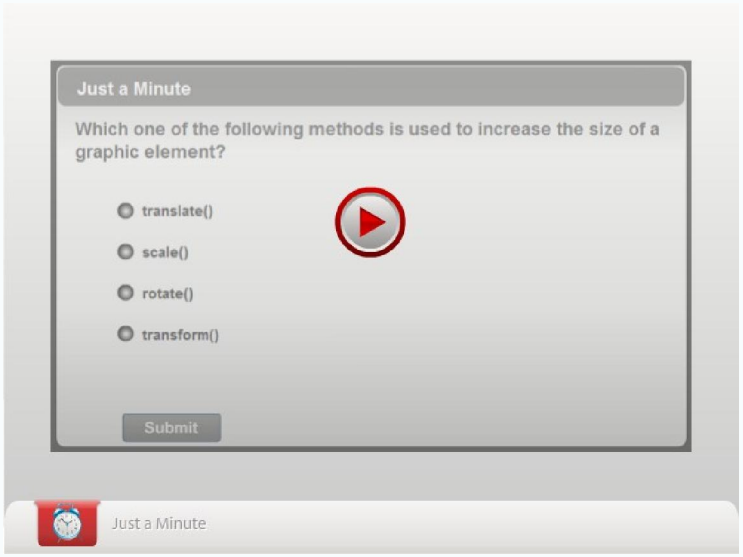
```
<!DOCTYPE HTML>
<HTML>
<BODY>
<CANVAS ID="myCanvas"  width="300"
height="300" style="border:1px solid
black">
</CANVAS>
<SCRIPT>
var c=document.getElementById
("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="red";
ctx.fillRect(10,10,100,100);
ctx.rotate(25);
ctx.fillStyle="blue";
ctx.fillRect(40,40,100,100);
</SCRIPT>
</BODY>
```

```
</HTML>
```

In the preceding code, a red colored rectangle is drawn at the coordinate, (10, 10). Further, the angle to rotate the canvas element is specified as 25. Therefore, the rectangle will be redrawn in the blue color starting from the coordinate, (40, 40), and rotated at the 25 degree angle, as shown in the following figure.



The Output Derived by Using the rotate() Method



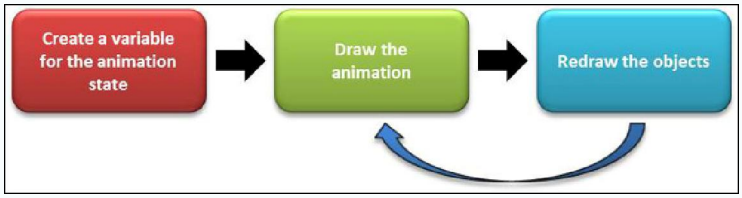
Animating Canvas Elements

An animation is a visual technique that provides the illusion that an object is moving by displaying graphic objects in rapid sequence. In an animation, the object is drawn first, next the canvas is cleared, and then the frame is drawn again with slight changes in its properties. The process is repeated very fast to create an illusion. For example, you want to represent a moving ball on the canvas. You have applied a single rotation on a ball that shows transformation. However, an animation is created if you apply multiple rotations on that ball.

To create such animations, you need to perform the following steps:

1. Create variables for the animation state.
2. Draw the animation.
3. Redraw the objects.

The following figure depicts the process of creating animations on the canvas.



The Process of Creating Animations on the Canvas

Creating Variables for the Animation State

Variables are created to store the initial properties of the graphic objects, such as width or height. Further, they can be used to store the updated values of the graphic objects. Therefore, whenever a new position is calculated for a graphic object, the new values will get stored in the variables.

Consider an example of creating a ball moving on a canvas surface. To create such an animation, you need to define the variables first. Consider the following code snippet for defining variables in JavaScript:

```
var canvas;
var ctx;
var x = 400;
var y = 300;
var dx = 2;
var dy = 4;
var WIDTH = 400;
var HEIGHT = 300;
```

In the preceding code snippet, the variables have been created that can be used to draw an animation on the canvas.

Drawing the Animation

After creating the variables, you need to draw the graphic object on the screen. Consider the following code snippet for drawing a ball on the canvas:

```
function circle() {
ctx.beginPath();
ctx.fillStyle="red";
ctx.arc(x, y, 10, 0, Math.PI*2,
true);
ctx.fill();
ctx.closePath();
}
function clear() {
ctx.clearRect(0, 0, WIDTH, HEIGHT);
}
function init() {
canvas = document.getElementById
("canvas");
```

```
ctx = canvas.getContext("2d");
return setInterval(draw, 10);
}
```

The preceding code snippet defines the following functions:

- `init()` function: In this function, the element ID for the canvas element is stored in the variable, `canvas`. Further, the context for the canvas element is retrieved in the variable, `ctx`. The context, `ctx`, can now be used to draw the graphic objects on the canvas. It is the first function that will be called in the code. In the next line of the code, the `setInterval()` function is used to call the `draw()` function every 10 milliseconds. The `draw()` function is a user-defined function that is created to redraw the ball on the canvas.

  
  You will study the code for the `draw()` function in the next topic.

- `circle()` function: In this function, the `beginpath()` method is called to start a new path. The `arc()` method is used to define the size and shape of the circle. Further, the `fill()` method is used to fill the entire circle with the specified color.
- `clear()` function: In this function, the `clearRect()` method is called to erase the graphic objects from the canvas.

Redrawing the Objects

Now, to create the animation effects, you need to repeatedly keep drawing the graphic objects with the updated properties.

Consider the following code snippet for redrawing a ball on the canvas:

```
function draw() {
 clear();
 circle();
 if (x> WIDTH || x< 0)
  dx = -dx;
 if (y> HEIGHT || y< 0)
  dy = -dy;
 x += dx;
 y += dy;
}
```

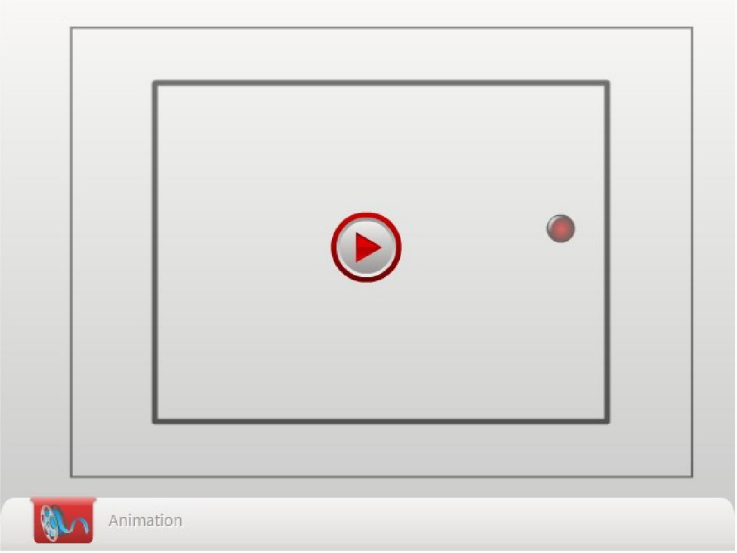
In the preceding code snippet, the `clear()` function is first called to clear the canvas. Further, the `circle()` function is called to create a circle on the canvas.

The circle has a radius, 10, and its origin is at `(x,y)`. To move the circle, you need to change the values of variables, x and y. Whenever the `draw()` function is executed, the variables, `dx` and `dy`, will determine the values for the variables, x and y. If the value of `x` is not greater than the width of the canvas or if it is less than zero, the value of x will get changed by dx. However, if the value exceeds the width size, then the variable, `dx` will be set to `-dx`, and the process continues. The same condition is applied for the variable, `y`.

The following lines show the entire code for rotating a ball:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<TITLE>Rotating Ball</TITLE>
</HEAD>
<BODY>
<DIV>
<CANVAS ID="canvas" width="400"
height="300" style="border:2px solid
black">
</CANVAS>
</DIV>
<SCRIPT type="text/javascript">
var canvas;
var ctx;
var x = 400;
var y = 300;
var dx = 2;
var dy = 4;
var WIDTH = 400;
var HEIGHT = 300;
function circle() {
  ctx.beginPath();
ctx.fillStyle="red";
  ctx.arc(x, y, 10, 0, Math.PI*2,
true);
  ctx.fill();
ctx.closePath();
}
function clear() {
  ctx.clearRect(0, 0, WIDTH, HEIGHT);
}
function init() {
  canvas = document.getElementById
("canvas");
  ctx = canvas.getContext("2d");
  return setInterval(draw, 10);
}
function draw() {
  clear();
  circle();
  if (x> WIDTH || x< 0)
    dx = -dx;
  if (y> HEIGHT || y< 0)
    dy = -dy;
  x += dx;
  y += dy;
}
init();
</SCRIPT>
</BODY>
</HTML>
```

The following file displays the output of the preceding code.



Activity 6.2: Creating a Game

Summary

In this lesson, you learned that:

- Canvas provides an easy and a powerful way to create graphics on a Web page.
- A canvas is defined by using the <CANVAS> tag. This tag is defined in the body section of the HTML document.
- Defining the canvas element only creates a blank drawing surface. However, to actually draw graphic objects on the canvas, you need to access the canvas in the JavaScript code.
- You can use the following methods to draw shapes on canvas:
  - `rect()`
  - `fillRect()`
  - `strokeRect()`
  - `clearRect()`
- The following properties can be used to apply colors on the canvas objects:
  - `fillStyle`
  - `strokeStyle`
  - `shadowColor`
- Apart from creating simple shapes on the canvas, you can also apply styles, such as gradients, on them.
- A path is a series of points joined together to create lines or shapes. In a canvas, you can use lines or paths to draw shapes other than rectangles or squares.
- In addition to drawing shapes or lines on the canvas, you can also draw text on it. You can also apply different styles on the text.
- The `drawImage()` method is used to draw an image or a video on the canvas.
- In a canvas, you can create graphs, such as bar graph or pie chart, to represent relationships.
- To transform the canvas elements, you can use the following methods:
  - `translate()`
  - `scale()`
  - `rotate()`
- To create such animations, you need to perform the following steps:
  - Create a variable for the animation state.
  - Draw the animation.
  - Redraw the objects.

Reference Reading

Introducing Canvas

Reference Reading: Books	***Reference Reading: URLs***
HTML5 Canvas Native Interactivity and Animation for the Web By Steve Foltun and Jeff Foltun	*http://sixrevisions.com/html/canvas-element/* *http://www.w3schools.com/html/html5_canvas.asp*

Transforming and Animating Canvas Elements

Reference Reading: Books	***Reference Reading: URLs***
HTML5 Canvas Native Interactivity and Animation for the Web By Steve Foltun and Jeff Foltun	*http://developer.apple.com/library/safari/#documentation/AudioVideo/Conceptual/HTML-canvas-guide/Translation,Rotation,andScaling/Translation,Rotation,andScaling.html* *http://tutorials.jenkov.com/html5-canvas/transformation.html*

Chapter 7

Adding Visual Effects to Web Pages

Today, almost every Web developer wants to add visual effects to websites. However, adding complex visual effects to Web pages requires writing many lines of JavaScript code, which is a time-consuming activity. To overcome this issue, jQuery was introduced. It is the JavaScript library supported by HTML that wraps several lines of code into methods that can be called by using a single statement.

This chapter introduces you to jQuery. Further, it discusses jQuery selectors, events, and effects. Also, it discusses how to add the image rollover effect to an image and how to create an image gallery, thereby, making the website more attractive and enhancing the browsing experience of the users.

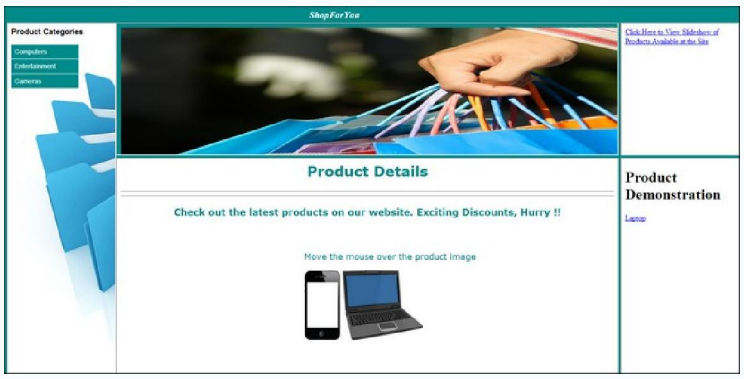
Objectives

In this chapter, you will learn to:

- Explore jQuery
- Add visual effects using jQuery
- Implement image rollover
- Create image gallery

Exploring jQuery

Consider a scenario of the online shopping website, ShopForYou.com. The website should be designed in such a way that it displays the product categories and items in the sliding and collapsible menu and submenu format, as shown in the following figure.



The Sample Web Page

For this, the Web designer has been asked to complete the task quickly and keep the code simple. For this situation, the designer can use the prewritten JavaScript library known as *jQuery*.

jQuery is an open source and cross-browser library of JavaScript codes that was created to make the JavaScript programming simpler. jQuery can be used by all levels of programmers to enhance the look and feel of the Web page. jQuery supports events and can be bound to the HTML elements on the Web page. jQuery bridges the gap between extensive coding and Web designing as it is easy to understand and provides many predefined special effects. With jQuery, a Web designer can add a plethora of impressive visual effects and also extend interactivity on a Web page.

Manipulating HTML Elements by Using jQuery

You can make use of jQuery in a Web page to manipulate HTML elements by downloading the light-weight jQuery JavaScript library and saving it to your system. Once the jQuery JavaScript library is downloaded, it can be referred to in a Web page by using the `<SCRIPT>` tag in the head section of the Web document.

The following syntax is used to specify the jQuery library:

```
<SCRIPT type="text/javascript"
src="<path>/jquery-1.8.3.js"> </
SCRIPT>
```

In the preceding syntax:

- The `SCRIPT` tag instructs the browser that the HTML document uses a script.
- The `type` attribute specifies the type of scripting used.
- The `src` attribute specifies the name of the jQuery library used. If the jQuery library is stored at the same location as the HTML document, only the name of the library can be given. However, if the jQuery library and the HTML document are stored at different locations, you have to specify the complete path of the jQuery library.

The execution of jQuery code should occur only after the Web page has been fully loaded. To ensure that a Web page is fully loaded before the jQuery code is executed on it, jQuery provides the `document.ready()` function. This function contains the jQuery code to be executed on HTML elements after the Web page has been fully loaded in the browser.

The following syntax is used to specify the `document.ready()` function:

```
<SCRIPT type="text/javascript">
$(document).ready(function(){
// jquery code...
});
</SCRIPT>
```

In the preceding syntax, the dollar sign ($) represents start of the jQuery code block and `jQuery code` is the code to be implemented for the rendering of the HTML elements in the browser window. The `document.ready()` function is useful for preventing failure of actions that should be performed on an HTML element. For example, if the jQuery code for hiding an image executes before the document is ready, it will lead

to failure. Therefore, to allow execution of the jQuery code only when the document is ready to be displayed in the browser window, all the jQuery methods and code should be declared and defined inside the `document.ready()` function.

The keyword, `jQuery`, can be used in place of the $ sign while writing jQuery code. For example, the preceding syntax can also be written as:

```
jQuery(document).ready(function(){
//jQuery code...
});
```

As jQuery is used to manipulate HTML elements, the elements should be selected first in order to implement the jQuery code on them.

The following jQuery syntax is used to select the HTML elements and perform the required action on them:

```
$(selector).action();
```

In the preceding syntax, `selector` is the element to be manipulated and `action` represents action to be taken on the selected element. jQuery provides many built-in actions that can be applied to the HTML elements, such as hide, fadeout, show, and slideup. Consider the following code to hide all the `<H1>` elements in an HTML document:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript"
src="jquery-1.8.3.js">
</SCRIPT>
<SCRIPT>
$(document).ready(function(){
$("h1").hide();
});
</SCRIPT>
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

In the preceding code snippet, all the `<H1>` elements will be hidden in the Web page.

jQuery code is written in the <HEAD> section of the document.

Using jQuery Selectors

jQuery uses selectors to select and manipulate an HTML element or a group of elements. You can select an HTML element by:

- Using element name
- Using attribute name
- Using CSS selectors

Using Element Name

jQuery can be used to select an HTML element or a group of HTML elements by using its name. Using jQuery, HTML elements can be selected by referring to their name, ID, or class to which they belong.

Consider the following code snippet to select all the `<H1>` elements in a document:

```
$("h1")
```

In the preceding code snippet, the `<H1>` elements are referred to by the name, `h1`. However, HTML elements can also be selected by using class or ID. To select an element with a specific ID by using jQuery, you can use the following syntax:

```
$("#Element_ID")
```

In the preceding syntax, `Element_ID` represents ID of the element to be selected. For example, to select an element having the ID, `header`, you can use the following code snippet:

```
$("#header")
```

To select elements with a specific class, you can use the following syntax:

```
$(".Element_class")
```

In the preceding syntax, `.Element_class` represents the class of the elements to be selected. For example, to select elements having the value of the class attribute as `header`, you can use the following code snippet:

```
$(".header")
```

For example, you have created the class selector named `exClass` and the ID selector named `exId` on the `<H1>` tag, as shown below:

```
<H1 class="exClass">
<H1 ID="exId">
```

Now, you want to manipulate the `<H1>` tag with the specified class and ID. You need to use the following code:

```
$("h1.exClass")
$("h1#exId")
```

Here, by using the `$("h1.exClass")` statement, only the `<H1>` header elements specified with the `exClass` value for the class attribute will be selected and by using the `$("h1#exId")` statement, only the `<H1>` header elements that are specified with the `exId` value for the ID attribute will be selected.

jQuery also supports nesting of elements. This allows applying an action on the elements that are nested within other elements. For example, a list item is nested inside a list. Consider the example of the ShopForYou online shopping website where the Web designer wants to implement a menu and submenu. The submenu items must be linked to display their details in a new document.

A menu comprises a list and list items that are denoted by the `<UL>` and `<LI>` tags, respectively. Consider the following code snippet to build a button, menu, and submenu, and to attach links to the submenu items:

```
<BODY>
<P> Product Menu</P>
<H4>Product Categories</H4>
<UL ID="menu">
<LI><A>Laptops</A>
<UL>
```

```
<LI><A href="">L-150</A></LI>
<LI><A href="">L-160</A></LI>
<LI><a href="">L-180</A></LI>
<LI><A href="">L-260</A></LI>
<LI><A href="">L-678</A></LI>
</UL>
</LI>
<LI><A>Phones</A>
<UL>
<LI><A href="">P-1150</A></LI>
<LI><A href="">P-1260</A></LI>
<LI><A href="">P-1180</A></LI>
<LI><A href="">P-2260</A></LI>
<LI><A href="">P-6708</A></LI>
</UL>
</LI>
<LI><A>Cameras</A>
<UL>
<LI><A href="">C-50</A></LI>
<LI><A href="">C-60</A></LI>
<LI><A href="">C-80</A></LI>
<LI><A href="">C-60</A></LI>
<LI><A href="">C-08</A></LI>
</UL>
</LI>
<LI><A>Music Players</A>
<UL>
<LI><A href="">MP-1150</A></LI>
<LI><A href="">MP-1260</A></LI>
<LI><A href="">MP-1180</A></LI>
<LI><A href="">MP-2260</A></LI>
<LI><A href="">MP-6708</A></LI>
</UL>
</LI>
</UL>
<BUTTON>Hide List</BUTTON>
</BODY>
```

Consider the following code snippet to hide all the nested `<UL>` items on the button click of a user, for which `menu` is specified as the ID attribute value:

```
<HEAD>
<SCRIPT src="jquery-1.8.3.js">
</SCRIPT>
<SCRIPT>
$(document).ready(function(){
$("button").click(function(){
$('#menu ul').hide();
});
});
</SCRIPT>
</HEAD>
```

In the preceding code snippet, `#menu` is used to select the HTML element for which the value of the ID attribute is `menu`. `#menu ul` selects all the `<UL>` items nested inside the HTML element for which the ID is specified as `menu`. When the user clicks on the button, all the nested list items will get hidden, and the menu with items at the first level: Laptops, Phones, Cameras, and Music Players will be visible.

Using Attribute Name

The HTML elements can also be selected by using the name of an attribute. The following syntax is used to select elements by using an attribute name:

```
$("[attribute-name]")
```

In the preceding syntax, `attribute-name` specifies the name of the attribute.

Consider the following code snippet:

```
$("[src]")
```

In the preceding code snippet, all the elements with the `src` attribute are selected.

Consider the following code snippet:

```
$("[src='movie.jpg']")
```

In the preceding code snippet, all the elements that have the `src` attribute as `movie.jpg` are selected.

Using CSS Selectors

jQuery provides CSS selectors to modify the CSS properties of an HTML element or document.

The following syntax is used to apply CSS selectors to modify the document or element properties:

```
selector.css( PropertyName,
PropertyValue );
```

Consider the following code snippet to change the color of the text of the `<DIV>` element to green:

```
$("div").css("color","green");
```

In the preceding code snippet, all the `div` elements are selected in the document, and the css `color` property value is modified to green.

With CSS selectors, you can set multiple properties on an HTML element. The following syntax is used to set multiple CSS properties:

```
css
({"propertyname":"value","propertynam
e":"value",...});
```

Consider the following code snippet to change the color and font size of the text of the `<DIV>` element:

```
$("div").css("color":"green", "font-
size":"200%");
```

In the preceding code snippet, all the `div` elements are selected in the document, and the css color property value is modified to green and font-size is set to 200%.

jQuery HTML

The jQuery library provides predefined functions to perform modifications on the content of HTML elements.

The following table lists the most commonly-used functions to modify the content of HTML elements.

Function	***Description***	***Syntax***
`html()`	*Changes the content of HTML elements, such as text in the `<P>` and `<DIV>` tags.*	*`$(selector).html (content)`*

append()	Enables a user to append content after the inside content in the selected element.	$(selector).append(content)
prepend()	Enables a user to append content before the inner content of selected elements.	$(selector).prepend(content)
remove()	Removes the selected element including all the text and nested elements inside it.	$(selector).remove()
empty()	Removes all the text and nested elements of selected elements. However, this method does not remove the element.	$(selector).empty()
after()	Inserts the content after the selected elements.	$(selector).after(content)
before()	Inserts the content before the selected elements.	$(selector).before(content)
text()	Sets or returns the content of selected elements.	$(selector).text(content)
val()	Sets or returns the values of the attributes of selected elements.	$(selector).val(value)
attr()	Sets or returns the values and attributes of selected elements.	$(selector).attr(attribute,value)

The jQuery HTML Functions

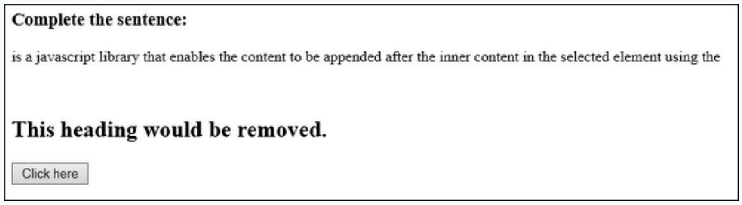
Consider the following code for modifying the inner content of HTML elements:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript"
src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$("button").click(function(){
  $("#para").prepend(" <B>jQuery </
B>");
  $("#para").append(" <B>append()
function</B>.");
  $("#para").after('<H3> This is the
newly inserted heading.</H3>');
  $("#newhead").remove();
  });
});
</SCRIPT>
</HEAD>
<BODY>
<DIV align="center">
</DIV>
<H3>Complete the sentence:</H3>
<P ID="para">is a javascript library
that enables the
content to be appended after the
inner content in the
selected element using the</P><BR>
<H2 ID="newhead">This heading would
be removed.</H2>
<BUTTON ID="b">Click here</BUTTON>
</BODY>
</HTML>
```

In the preceding code, when a user clicks the **Click here** button, the content, **jQuery,** is appended before the `<P>` tag having the ID, `para`, by using the `prepend()` function. Similarly, the content, `append()` function, is appended after the same `<P>` tag by using the `append()` function.

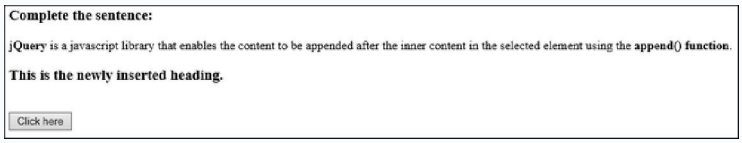
In addition, a new `<H3>` tag is inserted after the `<P>` tag having the ID, `para`, by using the `after()` function. Finally, the heading having the ID, newhead, is removed by using the `remove()` function.

Before the **Click here** button is clicked, the Web page appears, as shown in the following figure.



The Web Page Before the Button is Clicked

After the **Click here** button is clicked, the Web page appears, as shown in the following figure.



The Web Page After the Button is Clicked

The jQuery events are covered in the following topic.

Handling jQuery Events

In jQuery, events are handled by using functions or predefined event methods. An event method is used to detect an event and trigger a function when that event occurs. You can manipulate an element with a function

or an event method.
The following table lists some of the event methods provided by jQuery.

Event Method	*Description*
$(document).load (function)	*Used to execute a function after the document has finished loading.*
$(selector).click (function)	*Used to execute a function on the click event of the selected element.*
$(selector).dblclick(function)	*Used to execute a function on the double-click event of the selected element.*
$(selector).mouseenter(function)	*Used to execute a function when the mouse pointer is moved over the selected element.*
$(selector).mouseleave(function)	*Used to execute a function when the mouse pointer is moved out of the selected element.*
$(selector).keydown(function)	*Used to execute a function when a key is pressed on the selected element.*
$(selector).submit (function)	*Used to execute a function when a form is submitted.*
$(selector).focus (function)	*Used to execute a function when the selected element gets focus.*
$(selector).blur (function)	*Used to execute a function when the selected element loses focus.*
$(selector).resize (function)	*Used to execute a function when the browser window changes size.*
$(selector).unload (function)	*Used to execute a function when the user navigates away from the page.*
$(selector).mousedown(function)	*Used to execute a function when the left mouse button is pressed down on the selected element.*
$(selector).mouseup(function)	*Used to execute a function when the left mouse button is released from the selected element.*

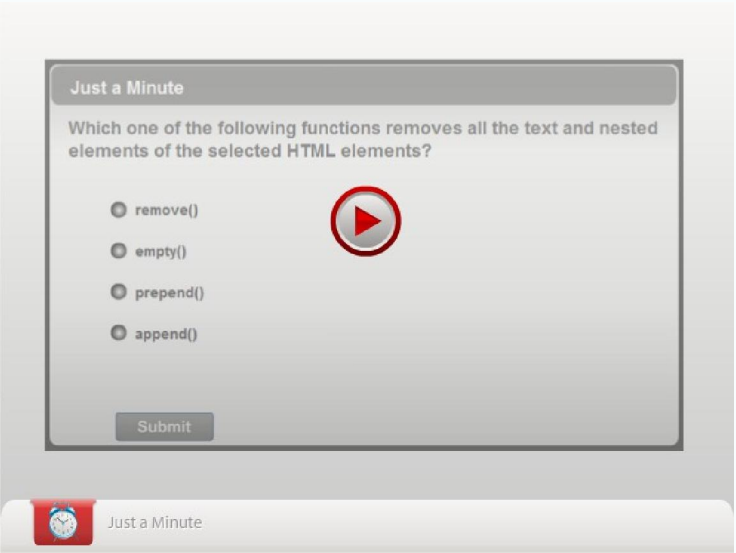
The Event Methods in jQuery

Consider the following code to handle jQuery events:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript"
src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$("#hide_header").click(function(){ /
* selects the hide_header button and
binds it to the click function */
$("h1").hide(); //hides the <h1>
header
});
});
</SCRIPT>
</HEAD>
<BODY>
<H1>I am the header</H1>
<HR>
<INPUT type="button" ID="hide_header"
value='Hide the header'/>
</BODY>
</HTML>
```

In the preceding code, jQuery recognizes a click event when the user clicks the **Hide the header** button. After the event is recognized by jQuery, the function associated with the button is executed and the `<H1>` element is hidden.

There are some predefined functions that can be attached to event methods in jQuery to enrich the visual appearance of a Web document. These functions are referred to as jQuery effects.



Adding Visual Effects Using jQuery

With the help of jQuery, amazing visual effects can be applied to the elements of a Web document. These visual effects help to enrich the browsing experience of a user. Some of the predefined jQuery effects that can be used to add visual appeal to a Web page are:

- Hide
- Show
- Toggle
- Slide
- Fade
- Animate

Implementing Hide Effect

The `hide()` function is used to make an element disappear when an event, such as click or double-click, occurs.

The following syntax is used to hide the selected elements:

```
$(selector).hide(speed)
```

In the preceding syntax, `speed` is an optional parameter that denotes speed with the values, slow, normal, and fast, or the duration in milliseconds at which an element disappears. Consider the following code to implement the hide effect:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript"
src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$("button").click(function(){
$("h1").hide("slow");
});
});
</SCRIPT>
</HEAD>
<BODY>
<H1>I am the header</H1>
<BUTTON>Hide the header</BUTTON>
</BODY>
</HTML>
```

In the preceding code, the text, **I am the header**, is displayed above the **Hide the header** button. When the **Hide the header** button is clicked, the text, **I am the header,** slowly disappears from the browser window.

Implementing Show Effect

The `show()` function is used to make a hidden element visible when an event occurs. The following syntax is used to show the selected elements, if they are already hidden:

```
$(selector).show(speed)
```

In the preceding syntax, `speed` is an optional parameter. This parameter is used to specify the speed with the values, slow, normal, and fast, or the duration in milliseconds at which an element reappears.

Consider the following code snippet to implement the show effect:

```
<!DOCTYPE HTML><HTML><HEAD>
<SCRIPT type="text/javascript"
src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$(".view").click(function(){
$("h3").show(2000);
});
$(".conceal").click(function(){
$("h3").hide();
});
});
</SCRIPT>
</HEAD>
<BODY>
<H3> This text can be hidden and
shown. </H3> <BR>
<BUTTON class="view">Click to view</
BUTTON>
<BUTTON class="conceal">Click to
hide</BUTTON>
</BODY>
</HTML>
```

In the preceding code snippet, the `<H3>` tag gets hidden in the browser window when the **Click to hide** button is clicked. When the **Click to view** button is clicked, the `<H3>` tag is shown again in 2000 milliseconds (or 2 seconds).

Implementing Toggle Effect

The `toggle ()` function can be used to switch between the show and hide effects of an element. This event can be used to hide or show the element, alternatively, when an event occurs.

The following syntax is used to show or hide the selected elements:

```
$(selector).toggle(speed)
```

In the preceding syntax, `speed` is an optional parameter. It is used to specify the speed with the values, slow, normal, and fast, or the duration in milliseconds at which an element disappears and reappears.

Consider the following code snippet to implement the toggle effect:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript"
src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$(".view").click(function(){
$("h3").toggle('fast');
});
});
</SCRIPT>
</HEAD>
<BODY>
<H3> Show and Hide me Quickly</H3>
<HR>
<BUTTON class="view">Click here</
BUTTON>
</BODY>
</HTML>
```

In the preceding code snippet, the `toggle()` function hides and shows the `<H3>` tag, alternately.

Implementing Slide Effect

The slide effect can be used to produce a sliding effect

on the selected elements. There are three slide functions, `slideDown()`, `slideUp( )`, and `slideToggle()`, which can be applied on the selected elements. These functions apply a gradual modification on the height of the selected elements to create the slide animation effect on a Web page.
The following table lists the slide effect functions.

Function	***Description***	***Syntax***
`slideDown()`	*Produces the effect of sliding selected elements in the downward direction.*	*`$(selector).slideDown (speed)`*
`slideUp()`	*Produces the effect of sliding selected elements in the upward direction.*	*`$(selector).slideUp (speed)`*
`slideToggle()`	*Performs the `slideup` and `slidedown` functions, respectively, on alternate clicks.*	*`$(selector).slideToggle (speed)`*

The Slide Effect Functions

Consider the following code to implement the slide effect:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript"
src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$(".flipbut").click(function(){
  $(".thought").slideToggle();
  });
});
</SCRIPT>
<STYLE type="text/css">
div.thought,.flipbut{ /*stylizing the
div element with the class
            attribute specified as
"thought",
            and the button with the
class attribute
            specified as "flipbut"
*/}
margin:0px;
padding:5px;
text-align:center;
background:beige;
border:solid 1px;
}
div.thought
{
height:120px;
display:none;
}
</STYLE>
</HEAD>
<BODY>
<BR><BR>
<BUTTON style="background-
color:beige" class="flipbut" text-
align="center">Show/Hide the thought
of the day</BUTTON>
<DIV class="thought">
<P>The thought of the day is:</P>
<P> Where there is a will, there is a
way!</P>
</DIV>
</BODY>
</HTML>
```

In the preceding code, when the `Show/Hide the thought of the day` button is clicked, toggle functionality is implemented between slide up and slide down on the `<DIV>` element along with the `thought` class.

For the online shopping website, ShopForYou.com, the menu and submenu must be sliding. When the user clicks a product category, the detailed products for that category should be displayed by using the slide toggle effect.

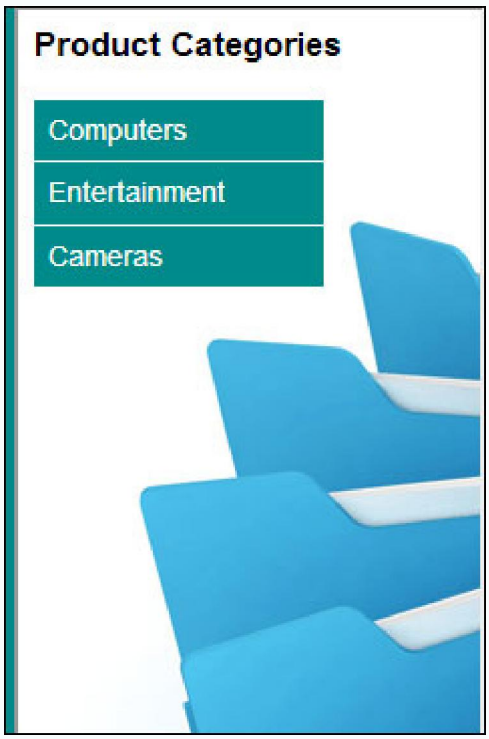
To implement this functionality in the ShopForYou.com website, you need to write the following code in a file and save the file as **menu.html**:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript"
src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$('#menu ul').hide();
  $('#menu li a').mouseenter(function
() {
    $(this).next().slideToggle
('slow');
    }
   );
});
</SCRIPT>
<STYLE type="text/css">
body{
background-image:url
('background.jpg');
}
li a
{
display:inline;
}
body {
font-family: Arial, sans-serif;
font-size: 0.9em;
background-color:silver;
}
```

```
p
{
  line-height: 1.5em;
}
#menu, ul#menu ul { /* stylizing the
menu and the list nested in the menu
*/
  list-style-type:none;
  margin: 0;
  padding: 0;
  width: 10em;
}
ul#menu a { /* stylizing the <a>
elements nested in the menu */
  display: inline;
  text-decoration: none;
}
ul#menu li { /* stylizing the list
items nested in the menu */
  margin-top: 1px;
}
ul#menu li a { /* stylizing the <a>
elements nested in list items of the
menu */
  background: darkcyan;
  color: #fff;
  padding: 0.5em;
}
ul#menu li a:hover { /* stylizing the
<a> elements nested in list items of
the menu when mouse is placed or
moved over them */
  background: black;
}
ul#menu li ul li a {
  background: grey;
  color: white;
  padding-left: 20px;
}
ul#menu li ul li a:hover {
  background: #aaa;
  color: white;
  border-left: 5px grey solid;
  padding-left: 15px;
}
.st
{
display:inline;
}
</STYLE>
</HEAD>
<BODY>
<H3>Product Categories</H3>
<UL ID="menu">
<LI class=".st"><A>Computers</A>
<UL>
   <LI><A href="">e-Book Readers</
A></LI>
   <LI><A href="">Tablets and i-
Pads</A></LI>
<LI><A href="">Laptops</A></LI>

</UL>
</LI>
<LI class=".st"><A>Entertainment</A>
<UL>
   <LI><A href="">LED-TVs</A></LI>
   <LI><A href="">LCD-TVs</A></LI>
</UL>
</LI>
<LI class=".st"><A>Cameras</A>
<UL>
   <LI><A href="">Digital Cameras</
A></LI>
   <LI><A href="">Digital SLR
Cameras</A></LI>
</UL>
</LI>
</UL>
</BODY>
</HTML>
```

In the preceding code, a menu that displays the **Computers, Entertainment, and Cameras** product categories is created and stylized by using CSS. A product category contains different product items under it. All product items are hidden and only the **Computers, Entertainment, and Cameras** product categories are visible, as shown in the following figure.



The Product Menu

When a product category is clicked, the product items that belong to that specific product category appear to slide down by using the slide toggle effect provided by jQuery. After clicking the **Entertainment** product category, a submenu is displayed, as shown in the following figure.

The Submenu for the Computers Product Category

When the product category is clicked again, the product items appear to slide up.

Implementing Fade Effect

The jQuery fade effect is used to gradually reduce the opacity of the selected elements. You can fade an HTML element in and out of visibility. The following table lists the four functions to produce the fade effect.

Function	***Description***	***Syntax***
`fadeOut()`	*Is used to fade the appearance of the selected elements.*	`$(selector).fadeOut(speed)`
`fadeIn()`	*Is used to restore the appearance of the faded, selected element.*	`$(selector).fadeIn(speed)`
`fadeTo()`	*Is used to specify the opacity for fading the selected element.*	`$(selector).fadeTo(speed,opacity)`
`fadeToggle()`	*Is used to fade the appearance of the selected elements, when they are faded out, and restore the appearance of the selected elements, when they are faded in.*	`$(selector).fadeToggle(speed)`

The Fade Effect Functions

Consider the following code for implementing the fade effect:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript"
src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
  $("div").click(function(){
  $(this).fadeOut(1200);
  });
$("button").click(function(){
  $("div").fadeIn(1200);
  });
});
</SCRIPT>
</HEAD>
<BODY>
<DIV
style="background:yellow;width:200px"
>FADE ME AWAY!</DIV><BR>
<BUTTON>Restore</BUTTON>
</BODY>
</HTML>
```

In the preceding code, the `div` tag is referred by using its name. In the click event handler of the div tag, the `this` keyword refers to the current HTML element, that is, the `div` tag. When the user clicks on the `div` tag, the text, `FADE ME AWAY!`, fades away. When the **Restore** button is clicked, the text, `FADE ME AWAY!`, reappears.

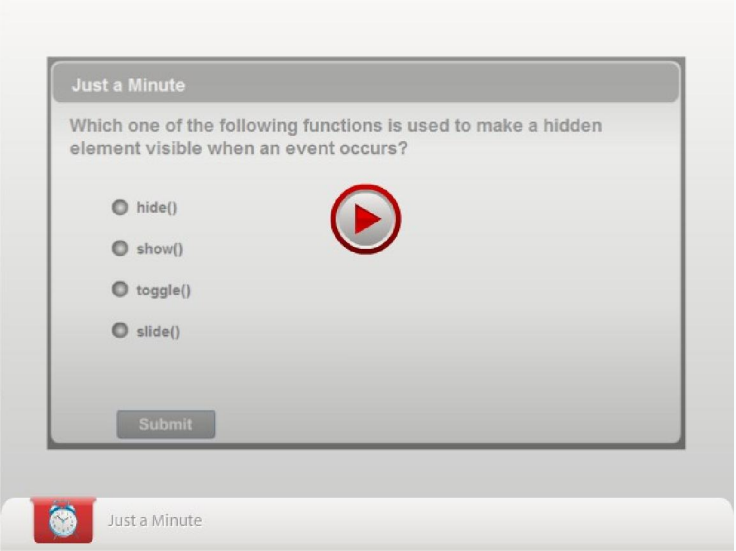
Implementing Animate Effect

The jQuery animate effect is used to create custom animations. You can create an animate function by using the following syntax:

`animate({params},speed,callback);`
where,
`params` specifies the CSS properties to be animated.
`speed` specifies the duration of the animation effect.
`callback` specifies the function that will be executed after completing the animation. It is an optional parameter.
Consider the following code snippet for applying animation effects on the `<DIV>` element:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT src="jquery-1.8.3.js">
</SCRIPT>
<SCRIPT>
$(document).ready(function(){
  $("div").mouseenter(function(){
  $("div").animate({
    center:'250px',
    opacity:'0.5',
    height:'250px',
    width:'250px'
  });
  });
$("div").mouseleave(function(){
  $("div").animate({
    opacity:'1',
    height:'100px',
    width:'100px'
  });
  });
});
</SCRIPT>
</HEAD>
<BODY>
<DIV
style="background:#98bf21;height:100p
x;width:100px;position:absolute;">
</DIV>
</BODY>
</HTML>
```

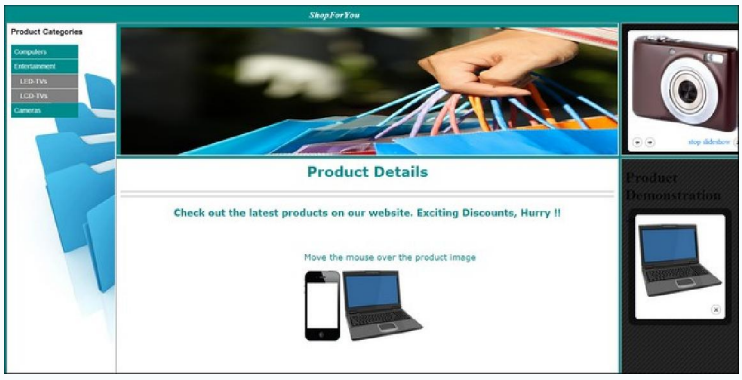
The preceding code will increase the size of the `<DIV>` element according to its specified width and height when the mouse is moved over it and will reduce the size of the `<DIV>` element according to its specified width and height when the mouse is left over it.



Activity 7.1: Exploring jQuery

Implementing Image Rollover

Consider the scenario of the ShopForYou.com website that sells products of various categories, such as phones and laptops. To give customers a preview of the appearance of these products, their images are displayed on the product page. To make the preview of these products appealing to the customers, you may want to add effects to the images displayed, such as when a customer places or moves the mouse pointer on an image, it appears enlarged or appears along with some information about that product. This can be achieved by replacing the image with a larger image of the same product or information about that product as plain text, as shown in the following figure.



The Sample Web Page

Creating Image Rollover

The images can be replaced by other images at runtime by creating the image rollover effect. JavaScript helps you to add the image rollover effect on Web pages.
Image rollover is an effect in which the appearance of an image changes when the mouse pointer is placed or

moved on it. The change in the appearance can be achieved by replacing the original image with another image by using JavaScript.

In the ShopForYou.com website, the **Product1.png** and**Product2.png** images display images of the laptop and mobile, respectively, on the product page. You need to add a feature on the website so that when a customer moves or places the mouse pointer over the image of the laptop or mobile, its details are displayed. To accomplish this task, you need to add the image rollover effect to the **Product1.png** and**Product2.png** images.

To implement the rollover effect, you need to write the following code in a file and save it as **imagerollover.html**:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<STYLE>
body{
background-color:white;
}
h2
{
font-family:verdana;
font-size:30px;
color:darkcyan;
text-align:center;
}
h3
{
text-align:center;
font-family:verdana;
color:darkcyan;
font-size:20px;
}
#para1
{
font-family:verdana;
font-size:18px;
text-align:center;
color:darkcyan;
}
</STYLE>
<SCRIPT>
function over(img,imgsrc)
{
img.src=imgsrc;
}
function out(img,imgsrc)
{
img.src=imgsrc;
}
</SCRIPT>
</HEAD>
<BODY>
<H2> Product Details </H2>
<HR><HR>
<H3> Check out the latest products on
our website. Exciting Discounts,
Hurry !!</H3>
<DIV style="position:absolute;
top:180px; left:400px;">
<P ID="para1"> Move the mouse over
the product image </P>
<IMG ID="productImage1"
src="Product1.png" height="150"
onmouseover="over
(this,'Product1Details.jpg')"
onmouseout="out
(this,'Product1.png')"/>
<IMG ID="productImage2"
src="Product2.png" height="150"
onmouseover="over
(this,'Product2Details.jpg')"
onmouseout="out
(this,'Product2.png')"/>
</DIV>
</BODY>
</HTML>
```

In the preceding code, two functions, `over()` and `out()`, are invoked on the `onmouseover` and `onmouseout` events of the `<IMG>` tag, respectively. The **Product1.png** and **Product2.png** images are added to the Web page by using the `<IMG>` tag. When the customer moves the mouse pointer on the **Product1.png** image, the `over()` function is invoked, which changes the value of the `src` attribute of the image in the Web page to **Product1Details.jpg**. As a result, the **Product1.png** image is replaced with the **Product1Details.jpg** image. However, when the customer moves the mouse away from the **Product1Details.jpg** image, the `out()`function is invoked, which changes the value of the `src` attribute of the image in the document to the **Product1.png** image. Therefore, the **Product1.png** image is redisplayed on the page. Similarly, the **Product2.png** image is replaced with the **Product2Details.jpg** image.

The following figure shows the output of the preceding code.

The Web Page Displaying the Images of Products

In the preceding Product Details Web page, if the mouse pointer is moved on the image of the laptop, it is replaced by another image that contains information of the product, as shown in the following figure.

The Web Page Displaying the Image Rollover Effect

Creating Backward Compatible Rollover

In most of the browsers, such as Firefox and Internet Explorer, the rollover effect can be created on images by using mouse events. However, in the earlier versions of browsers, mouse events on images cannot be captured. Therefore, the rollover effect cannot be added by using the mouse events on images in these browsers. To overcome this limitation, you need to create a backward compatible rollover.

For example, in the ShopForYou.com website, you need to add the rollover effect compatible for all browsers so that it can produce the desired output for all the customers, irrespective of the browser they are using.

The backward compatible rollover effect helps capture mouse events on images in all the browsers. To implement this effect, you need to use the `<A>` tag and place all the images by using the `<IMG>` tag within it. The `<A>` tag can capture mouse events, such as `onmouseover` and `onmouseout`, on the images.

Consider the following code snippet that creates the backward compatible rollover effect:

```
<A href="#" onmouseover="over()"
onmouseout="out()" >
<IMG ID="image1"
src="SampleImage1.gif">
</A>
```

In the preceding code snippet, the `<IMG>` tag is placed in the `<A>` tag. The `<A>` tag captures the mouse events on the SampleImage1.gif image. The captured events can be used to create the rollover effect.

In the preceding examples, you have used only the ID of the image with the `src` attribute to refer to a single image and replaced it with another image. However, to implement the rollover effect on multiple images, you can use the `document.images` array for easy implementation. The `document.images` array stores reference of all the images in the current document. If a reference of any image exists in the `document.images` array, the `document.images` array returns `true`. Therefore, the `document.images` array can be used to check if the images exist or not.

The various images in the `document.images` array can be referenced by using either the ID or the index of the image as a subscript of the array. For example, the first image in the document can be referred to by using the index of the image, as shown in the following code snippet:

```
document.images[0]
```

Similarly, the second image is referred to as `document.images[1]` and so on.

The images in an array can also be referenced by using the ID attribute of the image as a subscript of the array, as shown in the following syntax:

```
document.images [ID];
```

For example, in the ShopForYou.com website, to implement the rollover effect on multiple images and to ensure compatibility with all the browsers, you can use the following code on the product page of the ShopForYou.com website:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<STYLE>
h2
{
font-family:verdana;
font-size:30px;
text-align:center;
}
h3
{
font-family:verdana;
color:orange;
font-size:20px;
text-align:center;
}
#para1
{
font-family:verdana;
font-size:14px;
color:blue;
}
</STYLE>
<SCRIPT>
function over(img,imgsrc)
{
if(document.images)
{
document.images[img].src = imgsrc;
}
}
function out(img,imgsrc)
{
if (document.images)
{
document.images[img].src = imgsrc;
}
}
</SCRIPT>
</HEAD>
<BODY>
<H2> Product Categories </H2>
```

```
<HR><HR>
<H3> Check out the latest products on
our website. Exciting Discounts,
Hurry !!
</H3>
<DIV style="position:absolute;
top:180px; left:450px;">
<P ID="para1"> Choose the desired
product image to view the product
details
</P>
<A href="#" onmouseover="over
('productImage1','Product1Details.jpg
');" onmouseout="out
('productImage1','Product1.png');">
<IMG ID="productImage1"
src="Product1.png" border="0"/>
</A>
<A href="#" onmouseover="over
('productImage2','Product2Details.jpg
');" onmouseout="out
('productImage2','Product2.png');">
<IMG ID="productImage2"
src="Product2.png" border="0"/>
</A>
</DIV>
</BODY>
</HTML>
```

In the preceding code, the `<IMG>` tag defines the image source of the laptop and mobile, which are **Product1.png** and **Product2.png**, respectively. This tag is placed within the `<A>` tag. Therefore, the `<A>` tag enables you to capture mouse events on the **Product1.png** and **Product2.png** images. The `over()` and `out()` functions are invoked by using the `<A>` tag. The `over()` function is invoked when the user moves the mouse on an image. The `out()` function is invoked when the customer moves the mouse away from an image. Both these functions take two parameters, ID and name, of the image to be replaced. The `over()` and `out()` functions check the images in the document by using an array called `document.images`.

When the customer moves the mouse pointer on the **Product1.png** image, the `over()` function is invoked. This function replaces the **Product1.png** image with the Product1Details.jpg image by changing the value of the `src` attribute to the **Product1Details.jpg** image.

When the customer moves the mouse away from the image, the `out()` function is called to reset the value of the `src` attribute to the **Product1.png** image. Similarly, the **Product2.png** image can be replaced with the **Product2Details.jpg** image.

Preloading Images

Preloading images is a technique to load images in the browser cache before the script on the Web page is executed and the Web page is rendered in the browser window. This helps to avoid any delay in loading images from their respective locations, and then displaying these images on the Web page. The `Image` object can be used to preload images in an application. To use the `Image` object, an instance of the `Image` object needs to be created and the actual images need to be attached to the `Image` object in the head section. It ensures that the images are loaded in memory before the body of the page gets loaded.

The browser cache is used to store the downloaded files so that the files are not required to be downloaded again and instead can be used from the browser cache when the user revisits a page.

The following syntax is used to create an instance of the `Image` object:

```
var Imagename= new Image([Width],
[Height]);
```

In the preceding syntax:

- `Imagename`: Is the name of the new instance of the `Image` object.
- `[Width]`: Is the width of the image in pixels.
- `[Height]`: Is the height of the image in pixels.

While instantiating the `Image` object, both, the `[Width]` and `[Height]` parameters, are optional.

Consider the following code snippet that creates an instance of the `Image` object:

```
var img = new Image(40,30);
```

In the preceding code snippet, `img` is the name of the instance of the image object. The width and height of the image are assigned the values, `40` and `30`, respectively.

After instantiating an image object, you need to associate the image with the `Image` object. For this, the `src` attribute of the `Image` object is used. The syntax for attaching an image to an instance of the `Image` object is:

```
Imagename.src="ImageURL"
```

In the preceding syntax:

- `Imagename`: Is an instance of the `Image` object.
- `ImageURL`: Is the URL of the image.

Consider the following code snippet that attaches an image with the `Image` object:

```
img.src= SampleImage1.gif;
```

In the preceding code snippet, the `src` attribute of the `Image` object is used to attach the `SampleImage1.gif` image with the `img` object.

If an HTML document contains multiple images, you need to refer to all the images of the document while preloading images.

For example, in the ShopForYou.com website, you can use the `Image` object to preload the images and implement the rollover effect. For this, you can use the following code:

```
<!DOCTYPE HTML><HTML>
```

```
<HEAD>
<STYLE>
table{
text-align:center;
}
h2
{
font-family:verdana;
font-size:30px;
text-align:center;
}
h3
{
font-family:verdana;
color:orange;
font-size:20px;
text-align:center;
}
#para1
{
font-family:verdana;
font-size:14px;
color:blue;
text-align:center;
}
</STYLE>
<SCRIPT>
var images;
function preloadimages()
{
images = new Array
('product1.png','product2.png','produ
ct1details.jpg','product2details.jpg'
);
}
function over(a)
{
  document.images[a].src=images[a+2];
}
function out(a)
{
  document.images[a].src=images[a];
}
</SCRIPT>
</HEAD>
<BODY onLoad=preloadimages()>
<H2> Product Categories </H2>
<HR><HR>
<H3> Check out the latest products on
our website. Exciting Discounts,
Hurry !!</H3>
<DIV style="position:absolute;
top:180px; left:400px;">
<P ID="para1"> Move the mouse cursor
over the product image to view the
product details
</P>
<TABLE border="0">
<TR>
<TD>
<IMG src="product1.png"
onmouseover="over(0)" onmouseout="out
(0)" />
</TD>
<TD>
<IMG src="product2.png"
onmouseover="over(1)" onmouseout="out
(1)" />
</TD>
</TR>
</BODY>
</HTML>
```

In the preceding code, an array of images is created in the `preloadimages()` function. This array is passed as a parameter to the `preload()` function. In the `preload()` function, the images are preloaded by using the `Image` object. The images of the document are referred to by using the `document.images` array inside the `over()` and `out()` functions.

Creating Image Gallery

Consider the scenario of ShopForYou.com where an image gallery is used to display images of the products available at the website. An image gallery is a collection of thumbnail pictures or image links. All of these can be clicked individually to provide a large view of the corresponding product on another Web page. This image gallery is implemented by using JavaScript. A disadvantage of this feature is that to preview an image, the customer needs to navigate to the previous page repeatedly.

To avoid repeated navigation from a page to the previous page, you can display the full-size image on the same page. For this, you need to use jQuery plugins to create the image gallery. The jQuery plugins are prewritten and easy to use programs to enhance the jQuery code. By using jQuery plugins, you can create the image gallery by displaying the images dynamically at runtime on the same Web page. The jQuery plugins can be used to create an image gallery by:

- Using colorbox plugin
- Using galleria plugin

The light-weight, colorbox plugin enables displaying a collection of images one by one by using the previous and next buttons. The colorbox plugin can be used to display an image with a variety of transition effects, such as fade or elastic, in the middle of an elegant frame. This frame is known as lightbox and is created at the center of the Web page over a translucent film surrounding it. The colorbox plugin can be used effectively for creating and displaying an image gallery or a slideshow of images on the Web page where the image sequence can be controlled by using the previous and next buttons. To create an elegant image gallery or a slideshow on a Web page, you can download the **colorbox.zip** file that contains the necessary folders and files to implement the

colorbox plugin functionalities. Then, you can extract the colorbox folder from the **colorbox.zip** file. The colorbox folder comprises the following folders:

- **colorbox**: This folder comprises the **jquery.colorbox.js** file, which is the colorbox plugin file. This colorbox plugin file is used to implement the lightbox, slideshow, and the transition effects to display images in the image gallery.
- **example1**, **example2**,**example3**, **example4**, and **example5**: These folders contain sample implementation of the colorbox plugin. Each folder illustrates a unique and customized implementation of the colorbox plugin to create an image gallery or a slideshow. Each of these folders contains the following subfolders and files:
  - **colorbox.css**: This style sheet contains the formatting instructions to stylize, position, and decorate the image gallery as it is created by using the colorbox plugin. For example, background of the Web page, border of the image, and the previous, next, and close buttons.
  - **images folder**: This folder contains images that can be used to create and control the image gallery or slideshow. These images can be stylized and positioned on the Web page according to the instructions specified in the colorbox.css style sheet.
  - **index.html**: This Web page is used to illustrate how the image gallery can be displayed by using the elastic, fade, or no transition effects. In addition, this Web page is used to display a slideshow of the images.

Colorbox is a jQuery plugin. Therefore, to implement the functionality provided by this plugin, you need the `jquery-1.8.3.js` file. In addition, you need the jQuery event method, `document.ready()`, which is required to invoke the colorbox plugin to implement the slideshow or image gallery functionality.

Slideshow is used to display images one by one after a defined duration without using the previous and next buttons. However, in the image gallery, the user is required to use the previous and next buttons to scroll through all the images.

Let us consider an example to implement the colorbox plugin and create a lightbox for the image, **Product1.png**. For this, you need to perform the following steps:

1. Create a folder, such as **demo,** in any drive of your system.
2. Copy the downloaded **colorbox** folder to this folder.
3. Create the **jQueryS**, **CSS**, and**contentFiles** folders in the **demo** folder. This is done to ensure that the jquery file (**jquery-1.8.3.js)**, jquery plugin file (**jquery.colorbox.js)**, style sheet (**colorbox.css)**, and images (**Product1.png**) to be used are stored separately for easy implementation.

You can create the folder by any name of your choice. In fact, it is not necessary to create a new folder. This is done to ensure that all your related files are stored at one place. Depending on the storage location of your files, you need to specify the relative or absolute path of the respective files in the `src` attribute of the `<SCRIPT>` tag.

4. Copy the **jquery.colorbox.js** file from the **colorbox** folder available in the downloaded **colorbox** plugin folder to the**jQueryS** folder.
5. Copy the **colorbox.css** file from the **example1** folder available in the downloaded **colorbox** plugin folder to the**CSS** folder.
6. Copy the downloaded **jquery-1.8.3.js** jQuery file to the**jQueryS** folder.
7. Copy the **Product2.png** file to the **contentFiles** folder.
8. Copy the **images** folder from the **example1** folder available in the downloaded **colorbox** plugin folder to the **CSS** folder.
9. Type the following code in Notepad, and save the file as **LightboxDemo.html** at the same location where you have created the **jQueryS**, **CSS**, and**contentFiles** folders on your computer:

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
  <LINK media="screen"
rel="stylesheet" href="CSS/
colorbox.css" />
  <SCRIPT src="jQueryS/
jquery-1.8.3.js"></SCRIPT>
  <SCRIPT src="jQueryS/
jquery.colorbox.js"></SCRIPT>
  <SCRIPT type="text/
javascript">
    $(document).ready(function
(){
      $("a").colorbox();
    });
  </SCRIPT>
</HEAD>
<BODY>
  <H1>Product Demonstration</
H1>
  <A href="contentFiles/
Product2.png">Laptop</A>
</BODY>
</HTML>
```

In the preceding code, `a` represents the `<A>` element defined inside the `<BODY>` element. The `colorbox()` method of the colorbox plugin is used to display the **Product1.png** image inside the lightbox.

10. Open the **LightboxDemo.html** file in Microsoft Internet Explorer.

> *Click the **Allow blocked content** button if the following message is displayed:*
> ***Internet Explorer restricted this webpage from running scripts of ActiveX controls**.*

11. Click the **Laptop** hyperlink. The slide view of the product image is displayed with the default transition effect by using lightbox, as shown in the following figure.

The Lightbox Displaying Product Image

In the preceding figure, lightbox displays a single image with the default transition effects. However, you can customize the display of images in an image gallery or a slideshow by using the colorbox plugin. For this, the colorbox plugin provides several keys, which need to be passed as parameters to the `colorbox()` method. The following table lists the keys that can be used with the `colorbox()` method.

Key	***Description***
`transition`	*It specifies the type of transition, such as `fade`, `elastic`, or `none`, required for an image on the Web page. Its default value is `elastic`.*
`speed`	*It specifies the speed of transition with which the image is displayed. Its default value is `350 milliseconds`.*
`width`	*It specifies the total width of the frame in which the image is displayed. Its default value is `false`, which means that the width of the frame is not fixed and varies according to the width of the image.*
`height`	*It specifies the total height of the frame in which the image is displayed. Its default value is `false`, which means that the height of the frame is not fixed and varies according to the height of the image.*
`innerWidth`	*It specifies the inner width of the frame in which the image is displayed. This does not include the border and buttons.*
`innerHeight`	*It specifies the inner height of the frame in which an image is displayed. This does not include the border and buttons. Its default value is `false`.*
`current`	*It specifies the text that appears when multiple images are displayed by using lightbox, such as `image 1 of 5` when the first image in a group of five images is displayed. Its default value is `image {current} of {total}`. The {current} is detected and replaced with the actual number of images being displayed while ColorBox runs. {total} is detected and replaced with the actual number of images in the gallery.*
`slideshow`	*It specifies whether the slideshow of images needs to be created when multiple images are grouped together for preview. Its default value is `false`.*

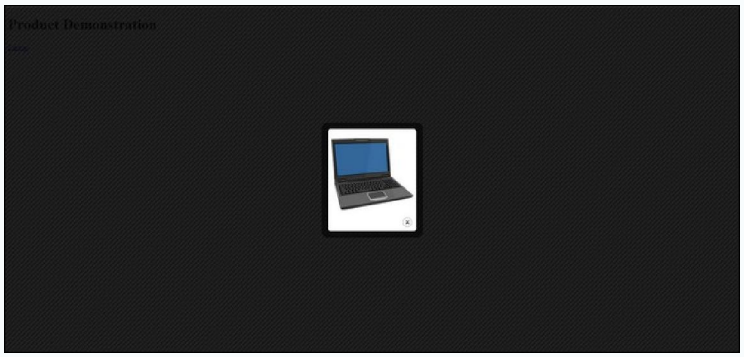
The Keys of the Colorbox Plugin

Consider the following code to create a lightbox by specifying the height and width of the frame in which the image is displayed:

```
<!DOCTYPE HTML>
```

```
<HTML>
  <HEAD>
  <LINK media="screen"
rel="stylesheet" href="CSS/
colorbox.css" />
  <SCRIPT src="jQueryS/
jquery-1.8.3.js"></SCRIPT>
  <SCRIPT src="jQueryS/
jquery.colorbox.js"></SCRIPT>
  <SCRIPT type="text/javascript">
    $(document).ready(function(){
  $("a").colorbox({transition:"fade",
height:"250", width:"220"});
    });
  </SCRIPT>
</HEAD>
<BODY>
  <H1>Product Demonstration</H1>
  <A href="contentFiles/
Product2.png">Laptop</A>
</BODY>
</HTML>
```

The preceding code creates a lightbox of height `250 px` and width `220 px` and displays the image inside this lightbox by using the `fade` transition. The output of the preceding code after the `Product1 photo` link is clicked is displayed, as shown in the following figure.



The Lightbox Displaying Product Image

You can also control the display of a set of images in an image gallery with the previous and next buttons. To display a set of images or a group of images by using a lightbox, you need to add a link to each image by using the `<A>` tag and specify the same value for the `rel` attribute of each `<A>` tag defined for the images. The `rel` attribute indicates the relationship between the current document and the path specified by the `href` attribute. By providing the same value for the `rel` attribute for all the images, these can be related to a single group.

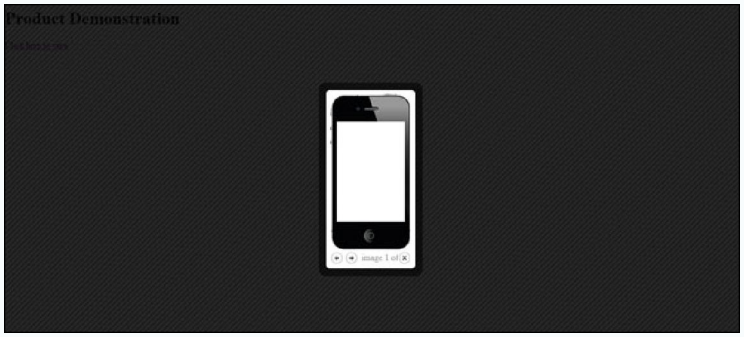
Suppose, you need to create a lightbox for the set of images, **Product1.png**, **Product2.png**, and **Product3.png**. For this, you need to perform the following steps:

1. Copy the **Product2.png** and **Product3.png** files to the **contentFiles** folder.
2. Type the following code in Notepad, and save the file as **LightboxGroupDemo.html** at the same location where you have created the **jQueryS**, **CSS**, and**contentFiles** folders on your computer:

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
  <LINK media="screen"
rel="stylesheet" href="CSS/
colorbox.css" />
  <SCRIPT src="jQueryS/
jquery-1.8.3.js"></SCRIPT>
  <SCRIPT src="jQueryS/
jquery.colorbox.js"></SCRIPT>
  <SCRIPT type="text/
javascript">
    $(document).ready(function
(){
      $("a").colorbox();
    });
  </SCRIPT>
</HEAD>
<BODY>
  <H1>Product Demonstration</
H1>
<A rel="collection"
href="contentFiles/
Product1.png">Click here to
view</A>
<A rel="collection"
href="contentFiles/
Product2.png"></A>
<A rel="collection"
href="contentFiles/
Product3.png"></A>
</BODY>
</HTML>
```

In the preceding code, the `rel` attribute of each `<A>` tag defined inside the body tag is specified with the value, `collection`. This ensures that the images having the same value for the `rel` attribute are grouped together so that these images can be displayed as part of a single image gallery.

When you save the preceding code in an HTML file, and then open it in Microsoft Internet Explorer, the **Click here to view** hyperlink is displayed. When you click the hyperlink, the output is displayed with the previous and next buttons, as shown in the following figure.



The Product Image with the Previous and Next Buttons

Click the ***Allow blocked content*** *button if the following message is displayed:*
Internet Explorer restricted this webpage from running scripts of ActiveX controls*.*

You can create a slideshow of images by grouping these images by using the colorbox plugin. For this, you need to use the same value for the `rel` attribute for all the images to be displayed in the `<A>` tag and set the `slideshow` key of the `colorbox()` method to true.

Consider the following code that creates a slideshow and saves it as **SlideShow.html**:

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
  <STYLE>
body{
Background-color: "silver";}
</STYLE>
  <LINK media="screen"
rel="stylesheet" href="css/
colorbox.css" />
  <SCRIPT src=" jQueryS/
jquery-1.8.3.js"></SCRIPT>
  <SCRIPT src=" jQueryS/
jquery.colorbox.js"></SCRIPT>
  <SCRIPT>
    $(document).ready(function(){
      $("a").colorbox
({slideshow:true,current:false});

    });
  </SCRIPT>
</HEAD>
<BODY>
<A href=" contentFiles/Product1.png"
rel="collection">Click Here to View
Slideshow of Products Available at
the Site
</A>
<A href=" contentFiles/Product2.png"
rel="collection"></A>
<A href=" contentFiles/Product3.png"
rel="collection"></A>
</BODY>
</HTML>
```

In the preceding code, the `slideshow` key is set to true to create a slideshow of the images. The value of the `current` key is set to `false`. This prevents displaying the text depicting the number of the current image, such as image 2 of 5, during the slideshow. The output of the preceding code is shown in the following figure.

The Slideshow of the Images

Using galleria Plugin

The galleria plugin is the jQuery plugin used for creating elegant and professional-looking image galleries. This plugin is used to display an image gallery and create thumbnails of the images to be displayed in the gallery automatically. When the mouse pointer is moved on a thumbnail image, a larger version of the image is displayed by using a transition effect on the same Web page. Unlike the colorbox plugin, the galleria plugin cannot be used to create a slideshow of images, but the galleria plugin has the ability to create thumbnails of the images to be displayed in the image gallery.

To create an image gallery by using the galleria plugin, you need to download the **galleria.zip** file from the Internet and extract the galleria folder from the **galleria.zip** file at a desired location on your system. The galleria folder comprises the **galleria-1.2.8.js** file, which is the galleria plugin to be used to implement the functionality of creating the image gallery.

Activity 7.2: Creating an Image Gallery

Summary

In this chapter, you learned that:

- jQuery is an open source and cross-browser library of JavaScript codes that was created to make the JavaScript programming simpler.
- You can make use of jQuery in a Web page to manipulate HTML elements by downloading the light-weight jQuery JavaScript library and saving it to your system.
- To ensure that a Web page is fully loaded before the jQuery code is executed on it, jQuery provides the `document.ready()` function.
- Using jQuery, HTML elements can be selected by referring to their name, ID, or class to which they belong.
- jQuery provides CSS selectors to modify the CSS properties of an HTML element or document.
- In jQuery, events are handled by using functions or predefined event methods.
- Some of the predefined jQuery effects that can be used to add visual appeal to a Web page are:
  - Hide
  - Show
  - Toggle
  - Slide
  - Fade
  - Animate
- Image rollover is an effect in which the appearance of an image changes when the mouse pointer is placed or moved on it.
- Preloading images is a technique to load images in the browser cache before the script on the Web page is executed and the Web page is rendered in the browser window.
- The `Image` object can be used to preload images in an application.
- An image gallery is a collection of thumbnail pictures or image links.
- By using jQuery plugins, you can create the image gallery by displaying the images dynamically at runtime on the same Web page.
- The light-weight, colorbox plugin enables displaying a collection of images one by one by using the previous and next buttons.
- The galleria plugin is the jQuery plugin used for creating elegant and professional-looking image galleries.

Reference Reading

Exploring jQuery

Reference Reading: Books	***Reference Reading: URLs***
JavaScript: the complete reference By Thomas A. Powell, Fritz Schneider	*http://www.w3schools.com/jquery/default.asp* *http://jquery.com/*

Adding Visual Effects Using jQuery

Reference Reading: Books	***Reference Reading: URLs***
JavaScript: the complete reference By Thomas A. Powell, Fritz Schneider	*http://api.jquery.com/category/effects/*

Implementing Image Rollover

Reference Reading: Books	***Reference Reading: URLs***
JavaScript: the complete reference By Thomas A. Powell, Fritz Schneider	*http://www.irt.org/articles/js132/*

Creating Image Gallery

Reference Reading: Books	***Reference Reading: URLs***
JavaScript: the complete reference By Thomas A. Powell, Fritz Schneider	*http://monc.se/kitchen/146/galleria-a-javascript-image-gallery*

Chapter 8

Introducing Geolocation and Offline Support for Data

You are visiting a city for the first time and looking for sight-seeing locations close to your hotel. However, you do not know where to go and which places to look for. Therefore, it will be helpful if you have some application that can help you locate such places and provide feedback for the same. This can be done by implementing Geolocation API.

In addition, you may need that users are able to access a Web page even when they are not connected to the Internet.

This chapter introduces you to the Geolocation API. Further, it discusses how to implement offline support in websites.

Objectives

In this chapter, you will learn to:

- Implement Geolocation
- Implement Offline Support

Implementing Geolocation

ShopHere is a large retail store based in Ohio. The store offers clothes, accessories, and footwear. In addition, it sells the home furnishing goods and electronic items, such as refrigerator, air conditioner, laptops, and mobile devices. As part of its business strategy to attract new customers and build a stable customer base, the store offers discounts and special deals on products. In addition, the official website of the store has a feature that enables users to share their current location, based on which, they are guided to reach the nearest store.

Such a feature is incorporated in websites by using the Geolocation API. The Geolocation API-enabled website can locate users' current geographical location, display points of interest around that location, or guide the users from their current location to a target destination.

Implementing the Geolocation API

The Geolocation API allows a website to retrieve the current geographical location of a user. This API enables you to create applications that guide users how to reach a target location from their current location. An example of such an application is a map application that gives directions to a target location.

The users' location is not retrieved just through code or browser. Instead, the Geolocation API uses certain features, such as Global Positioning System (GPS), Internet Protocol (IP) address of a device, nearest mobile phone towers, and input from a user, in the users' device to retrieve the users' location.

The users' location retrieved by using the Geolocation API is almost accurate depending upon the type of source used to retrieve the location. For example, the IP address gives a location that can be close to the users' location; while, the GPS is able to give more accurate results.

Identifying the users' location may, at times, compromise the users' privacy. Hence, the location of users is not available unless they approve it. A prompt appears that asks the users if they would like to share their current location and also specifies the reason for collecting this data. In addition, it should specify where the data will be shared.

Geolocation is most beneficial with applications used for mobile devices. This is because while the users are travelling, the Geolocation API keeps updating their locations. In case of desktop applications, the users' location remains constant and can be set only once.

The Geolocation API provides the following methods to determine the users' location:

- `getCurrentPosition()`
- `watchPosition()`

getCurrentPosition()

The `getCurrentPosition()` method is used to retrieve the current geographical location of a user. The location is retrieved as a set of coordinates.

The syntax of the `getCurrentPosition()` method is:

```
getCurrentPosition(CallbackFunction,
ErrorHandler, Options);
```

In the preceding syntax:

- `CallbackFunction`: This is a function defined by the developer to retrieve the current location.
- `ErrorHandler`: This is the name of a function that is called when an error occurs while retrieving the location of a user. This is an optional parameter.
- `Options`: This optional parameter specifies a set of options, such as timeout for retrieving the location information, used for retrieving the information about the users' geographical location.

The `getCurrentPosition()` method calls callbackFunction, which takes the `position` object as an argument. This object specifies the current geographic location of a user as a set of geographic coordinates. The `position` object returns two properties, `coords` and `timestamp`, on the successful retrieval of the location. The `timestamp` property returns the date and time when the location was retrieved. The `coords` property returns various attributes, such as `latitude` and `longitude`. These attributes are described in the following table.

Attribute	***Description***
`coords.latitude`	*Specifies the latitude as a decimal number.*
`coords.longitude`	*Specifies the longitude as a decimal number.*
`coords.accuracy`	*Specifies the accuracy of position.*
`coords.altitude`	*Specifies the altitude in meters above the sea level.*
`coords.altitudeAccuracy`	*Specifies the accuracy of altitude.*

The Attributes of the coords Property

Before retrieving the users' location by using the `getCurrentPosition()` method, you need to check whether the browser supports the geolocation feature. For this, you can use the `geolocation` property of the `navigator` object. The `navigator` object contains the information about a browser. Consider the following code snippet to check whether the browser supports the geolocation feature:

```
if(navigator.geolocation)
var geo=navigator.geolocation
else
{
alert("Your browser does not support
geolocation")
}
```

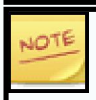
The preceding code snippet checks whether the geolocation feature is supported by the browser. If it is not supported, a relevant message is displayed.

Consider the following code snippet for retrieving the users' location by using the `getCurrentPosition()` method:

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<P ID="button">Click here to know
your location coordinates:</P>
<BUTTON onclick="getLocation()">Get
Location</Button>
<SCRIPT>
var geo=document.getElementById
("button");
function getLocation()
{
if (navigator.geolocation)
{
navigator.geolocation.getCurrentPosit
ion(getPosition);
}
else{geo.innerHTML="Geolocation is
not supported by this browser.";}
}
function getPosition(position)
{
geo.innerHTML="Latitude: " +
position.coords.latitude +
"<BR>Longitude: " +
position.coords.longitude;
}
</SCRIPT>
</BODY>
</HTML>
```

The preceding code snippet creates the **Get Location** button, which, on clicking, calls the `getLocation()` method. This method first checks whether the geolocation feature is supported by the browser. If this feature is supported, the `getCurrentPosition()` method is called. This method calls the `getPosition()` method that retrieves the users' location in the form of latitude and longitude values.

The innerHTML property is used to set or retrieve the text of an HTML element.



Due to security reasons and firewall restrictions, you may not always get the latitude and longitude coordinates.

watchPosition()

The `watchPosition()` method returns the current location of a user and continuously updates the location while the user is moving. It is mostly used in devices, such as GPS, which inform the users about the current location while they are travelling. The

watchPosition() method takes the same parameters as the getCurrentPosition() method and returns the same object.

Consider the following code snippet for retrieving the users' location by using the watchPosition() method:

```
function getLocation()
  {
  if (navigator.geolocation)
    {
    navigator.geolocation.watchPositi
on(getPosition);
    }
  else{geo.innerHTML="Geolocation is
not supported by this browser.";}
  }
function getPosition(position)
  {
  geo.innerHTML="Latitude: " +
position.coords.latitude +
  "<br>Longitude: " +
position.coords.longitude;
  }
```

The preceding code snippet continuously retrieves the user's location by using the watchPosition() method while the user is travelling.

You can also stop tracking the users' location by using the clearWatch() method. This method stops the call to the watchPosition() method.

Handling Errors

At times, the user may not provide the permission to access his location. This may raise an error in your website. The error may also occur when a user checks the current location on a mobile device and the device goes out of coverage area or the network connection is timed out. As a website developer, you need to ensure that your website is able to handle errors effectively. For this, you need to consider all possible causes of errors that can occur in an application and can be dealt with effective error-handling technique.

To handle errors in a geolocation-enabled website, you can use the getCurrentPosition() method. The second parameter of this method is an error handler function. This function can handle the following error codes:

- PERMISSION_DENIED: Specifies that the user has declined the request to share the location.
- POSITION_UNAVAILABLE: Specifies that the users' current location cannot be retrieved.
- TIMEOUT: Specifies that the time given to retrieve the users' location has exceeded the maximum limit.
- UNKNOWN_ERROR: Specifies that an unknown or undefined error has occurred.

The error handler function accepts the PositionError object as an argument. This object has two properties, code and message. The code specifies the error code, and the message specifies the message that is to be displayed when an error occurs.

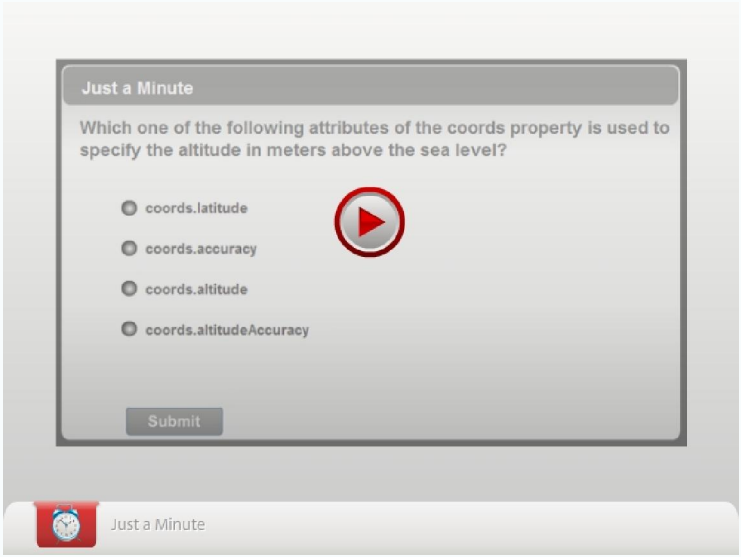
Consider the following code snippet for handling errors while retrieving the users' location:

```
function showError(PositionError)
{
switch(PositionError.code)
   {
   case
PositionError.PERMISSION_DENIED:
     x.innerHTML="User denied the
request for tracking the location"
     break;
   case
PositionError.POSITION_UNAVAILABLE:
     x.innerHTML="User's location is
not available"
     break;
   case PositionError.TIMEOUT:
     x.innerHTML="The request to
retrieve user's location is timed
out"
     break;
   case PositionError.UNKNOWN_ERROR:
     x.innerHTML="An unknown error
occurred"
     break;
   }



}
```

The preceding code snippet creates the function, showError(), which is called when an error occurs while retrieving the users' location. This function accepts an error code which is defined in the switch construct. For example, if the user denies the access to his location, the error,User denied the request for tracking the location, is displayed on a Web page.



Implementing Offline Support

At times, you must have noticed that while you are surfing a website, a message, such as 'Browser could not open the Web page', is displayed. This is because the connection to the Internet is lost. In such a case, you do not have access to the website that you were surfing. However, if you want to have some level of access to the website even if you are not connected to the Internet, you can use the offline support feature. For example, the executives of ShopHere often need to visit different cities to advertise their products. They need to access the organization's website during their promotional tours. However, the website is not accessible every time, especially in the remote areas where the network is not available. In such a case, the offline support feature of HTML can be used to ensure that the website is accessible to the executives even without the Internet connection. In addition, using the offline support feature avoids the normal network requests needed to load a website.

Before the offline support feature of HTML, the offline storage-enabled websites were created using cookies and plugins. A cookie is a small piece of data, which is sent from a website and stored in a user's browser while the user browses the website. When the user revisits the same website, the data stored in the cookie is retrieved. Cookies are not purely used for offline storage as they store a limited amount of data. In addition, cookies slow down the network activity because they are transferred to and from the server.

These limitations are overcome with the help of the offline support feature of HTML. The offline support feature provides the following benefits:

- Ensures that the website is available even when the user is not connected to the network.
- Reduces network load on the server.

You can make your website work offline by using the following ways:

- Implementing client-side storage
- Implementing application cache

Implementing Client-side Storage

The client-side storage refers to the process of storing data locally within the user's browser. This is also known as Web storage. The data stored by using client-side storage is retrieved only when it is requested, and not with every server request. Moreover, you can store a large amount of data by implementing the client-side storage, where the data is stored in the form of key/value pairs.

The client-side storage can be implemented by using the following objects:

- `localStorage`
- `sessionStorage`

localStorage

The `localStorage` object allows you to store data without any expiration date. This implies that the data stored by using the `localStorage` object is not deleted after the browser is closed, and it will be available when the browser is reopened. The `localStorage` object stores the data only in the form of string. Using this object, the cached data is accessible across all browser windows.

Consider the following code snippet for storing the users' information by using `localStorage`:

```
<DIV ID="str"></DIV>
<SCRIPT>
if(typeof(Storage)!=="undefined")
  {
  localStorage.name="John";
  document.getElementById
("str").innerHTML="Name: " +
localStorage.name;
  }
else
  {
  document.getElementById
("str").innerHTML="Your browser does
not support local storage";
  }
</SCRIPT>
```

In the preceding code snippet, the `typeof()` method first checks whether the browser supports the Web storage or not. If it does, the`localStorage` object stores the user's name locally in the browser by using the key, `name`, which is assigned the value, `John`.

Later, the information stored in the `localStorage` object is retrieved by using the key and is assigned to the `innerHTML` property of the `<DIV>` tag that has `ID`, `str`.

The text, `Your browser does not support local storage`, is displayed inside the `<DIV>` tag if the browser does not support the Web storage.

sessionStorage

The `sessionStorage` object is used to store the data for only one session. This implies that the data is deleted

once the user closes the browser. The data stored by using `sessionStorage` is confined to the browser for which it was created. Consider an example where you need to count the number of times a user has clicked a button in the current session. For this, you can use the`sessionStorage` object, as shown in the following code snippet:

```
<HEAD>
<SCRIPT>
function clickCounter()
{
if(typeof(Storage)!=="undefined")
  {
  if (sessionStorage.clickcount)
    {
    sessionStorage.clickcount=Number
(sessionStorage.clickcount)+1;
    }
  else
    {
    sessionStorage.clickcount=1;
    }
  document.getElementById
("btn").innerHTML="You have clicked
it" +



sessionStorage.clickcount + " time(s)
in this session.";
  }
else
  {
  document.getElementById
("btn").innerHTML="Your browser does
not support Web storage";
  }
}
</SCRIPT>
</HEAD>
<BODY>
<p><BUTTON onclick="clickCounter()"
type="button">Click Here</BUTTON></P>
<DIV ID="btn"></DIV>
<P>Click the button to see the
increase in counter.</P>
<P>Close the browser and try again,
the counter will be reset.</P>
</BODY>
```

The preceding code snippet creates the `clickCounter()` function that is called when the button is clicked.

Inside the function body, the `typeof()` method first checks whether the browser supports the Web storage or not. If it does, the `clickCount` key is used with the `sessionStorage` object to store the count or number of times the user has clicked the button during the current session. The value of the key, `clickCount`, is increased by one every time the user clicks the button. Since the value assigned to the key of the `sessionStorage` object is always a string, it is converted into a number before incrementing by one, by using the `Number()` method.

The most recent value stored inside the `sessionStorage` object is retrieved by using the key, `clickCount`, and is assigned to the `innerHTML` property of the `<DIV>` tag that has `ID`, `btn`, which is displayed before the user.

Implementing Application Cache

While browsing a website, you must have faced a situation when the network connection is lost and you click the Back button in the browser to view the previous page. However, you are not able to view that page as you are not connected to the Internet, and the browser did not cache the page properly. To view the previous page in such a situation, you need to reconnect to the Internet. To address such issues, while developing a website, you can specify the files the browser should cache so that even if you refresh the page when you are offline, you are able to view the page. This process of caching a website is known as application cache.

The application cache provides the following advantages:

- **Offline browsing**: Specifies that a website can be viewed even if the user is not connected to the Internet.
- **Speed**: Specifies that if the user requests the Web page, which is already there in the cache, it is retrieved from the cache and not from the server. Therefore, the loading of the Web page is faster as the network is not accessed as the connection to the server is not needed.
- **Reduced server load**: Specifies that the Web page, if cached, will always be made available from the cache unless the browser detects that the cache manifest has been updated on the server or the user has cleared the browser cache. Then, the browser downloads a new version of the manifest and the resources listed in the manifest. Therefore, the number of requests sent to the server are less; thereby, reducing the load on the server.

To implement application cache, you need to create a text file called manifest. This file contains a list of resources that needs to be cached for use when there is no network connectivity. The manifest file also contains the list of files or pages that should never be cached. You need to save the manifest file with the .appcache extension. The manifest file is divided into the following sections:

- **Cache**: Lists the files that need to be cached after they are downloaded for the first time.
- **Network**: Lists the files that should never be cached.
- **Fallback**: Specifies the task to be performed

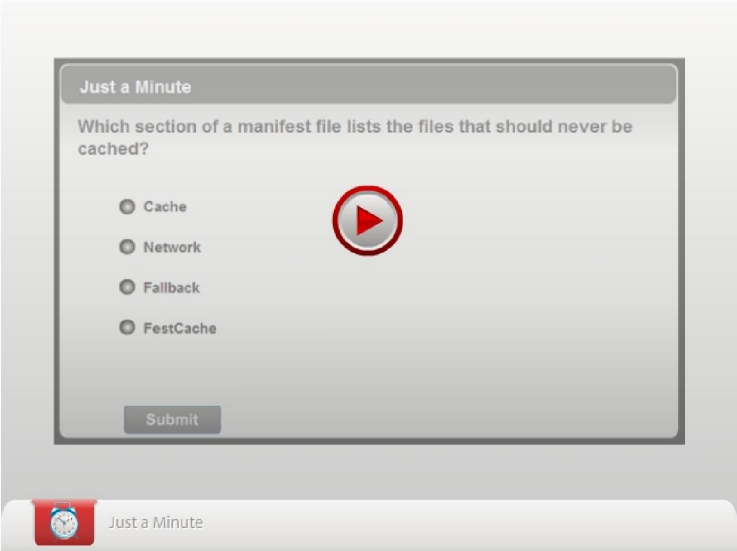
when a user tries to fetch the uncached files.

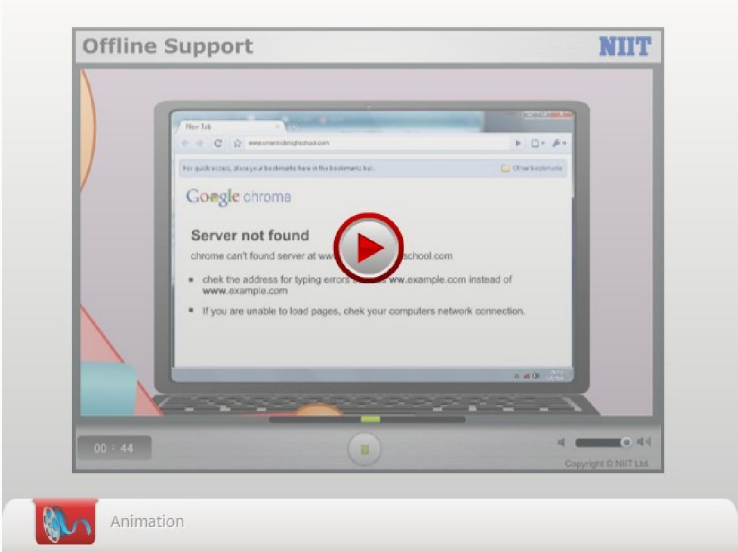
To refer to a manifest file in an HTML document, you need to add the manifest attribute in the opening<HTML> tag, as shown in the following code snippet:

```
<HTML
manifest="HotelFacilities.appcache">
```

The preceding code snippet enables the application cache for the `HotelFacilities.appcache` manifest file.

Once you have cached an application on the local machine, the browser shows the cached file even if it has been changed or updated on the server. Therefore, to ensure that the browser updates the cache, you need to modify the manifest file.





Activity 8.1: Implementing Offline Support

Summary

In this chapter, you learned that:

- The Geolocation API allows a website to retrieve the current geographical location of a user.
- The Geolocation API provides the following methods to determine the users' location:
  - `getCurrentPosition()`
  - `watchPosition()`
- To handle errors in a geolocation-enabled website, you can use the `getCurrentPosition()` method. The second parameter of this method is an error handler function.
- The offline support feature provides the following benefits:
  - Ensures that the website is available even when the user is not connected to the network
  - Reduces network load on the server
- The client-side storage refers to the process of storing data locally within the user's browser.
- The client-side storage can be implemented by using the following objects:
  - `localStorage`
  - `sessionStorage`
- The `localStorage` object allows you to store data without any expiration date.
- The `sessionStorage` object is used to store the data for only one session. This implies that the data is deleted once the user closes the browser.
- To implement application cache, you need to create a text file called manifest.
- You need to save the manifest file with the .appcache extension.

Reference Reading

Implementing Geolocation

Reference Reading: Books	*Reference Reading: URLs*
The Definitive Guide to HTML5 By Adam Freeman	*http://diveintohtml5.info/geolocation.html* *http://www.w3schools.com/html/html5_geolocation.asp*

Implementing Offline Support

Reference Reading: Books	*Reference Reading: URLs*
The Definitive Guide to HTML5 By Adam Freeman	*http://www.pageresource.com/html5/local-session-storage-api/*

Appendix

Browser Support for HTML5 Elements

HTML5 is a new standard for HTML. A lot of new features have been added to HTML5. However, all browsers currently do not provide support for all new features of HTML5.

HTML5 New Elements

The following table lists the browser support for the various elements introduced in HTML5.

Element Name	Internet Explorer	Firefox	Chrome	Safari	Opera
<ARTICLE>	✓	✓	✓	✓	✓
<ASIDE>	✓	✓	✓	✓	✓
<DETAILS>	X	X	✓	✓	X
<SUMMARY>	X	X	✓	✓	X
<FIGURE>	✓	✓	✓	✓	✓
<FIGCAPTION>	✓	✓	✓	✓	✓
<VIDEO>	✓	✓	✓	✓	✓
<DATALIST>	✓	✓	✓	X	✓
<KEYGEN>	X	✓	✓	✓	✓
<OUTPUT>	X	✓	✓	✓	✓
<METER>	X	✓	✓	✓	✓
<PROGRESS>	✓	✓	✓	✓	✓
<HGROUP>	✓	✓	✓	✓	✓
<NAV>	✓	✓	✓	✓	✓
<HEADER>	✓	✓	✓	✓	✓
<FOOTER>	✓	✓	✓	✓	✓
<AUDIO>	✓	✓	✓	✓	✓
<CANVAS>	✓	✓	✓	✓	✓

Support for Elements Introduced in HTML5

HTML5 Input Types

The following table lists the browser support for the various input type elements introduced in HTML5.

Element Name	Internet Explorer	Firefox	Chrome	Safari	Opera
date	X	X	✓	✓	✓
tel	X	X	X	X	X
time	X	X	✓	✓	✓
url	✓	✓	✓	X	✓
range	✓	X	✓	✓	✓
number	✓	X	✓	✓	✓
email	✓	✓	✓	X	✓

Browser Support for Input Type Elements Introduced in HTML5

HTML5 Form and Input Attributes

The following table lists the browser support for the various form attributes introduced in HTML5.

Attribute Name	Internet Explorer	Firefox	Chrome	Safari	Opera
novalidate	✓	✓	✓	X	✓
autocomplete	✓	✓	✓	✓	✓

The following table lists the browser support for the various form input type attributes introduced in HTML5.

Attribute Name	Internet Explorer	Firefox	Chrome	Safari	Opera
autofocus	✓	✓	✓	✓	✓
form	X	✓	✓	✓	✓
formaction	✓	✓	✓	✓	✓
formenctype	✓	✓	✓	✓	✓
formmethod	✓	✓	✓	✓	✓
formnovalidate	✓	✓	✓	X	✓
formtarget	✓	✓	✓	✓	✓
height	✓	✓	✓	✓	✓
width	✓	✓	✓	✓	✓
autocomplete	✓	✓	✓	✓	✓
min	✓	X	✓	✓	✓
max	✓	X	✓	✓	✓
multiple	✓	✓	✓	✓	✓
pattern	✓	✓	✓	X	✓
placeholder	✓	✓	✓	✓	✓
required	✓	✓	✓	X	✓
step	✓	X	✓	X	✓

Browser Support for Input Type Attributes Introduced in HTML5

The preceding table is based on support provided for HTML5 and CSS3 by the following popular browsers: Internet Explorer 10 Firefox 13.0.1 Opera 12.00 Google Chrome 20.0.1132.47 Safari 5.1.7 Please check for support provided for HTML5 and CSS3 in the version of

browser you are using.

Disclaimer: The support provided by various browsers for all the features of HTML5 listed in the preceding tables has been examined for accuracy and appropriateness at the time of addition. However, the same cannot be guaranteed over time as the browsers may start supporting more features of HTML5 in the near future.

Glossary

A

Animation

It is a visual technique that provides the illusion that an object is moving by displaying graphic objects in a rapid sequence.

Arithmetic Operator

It is used to perform arithmetic operations on variables and literals.

Assignment Operator

It is used to assign a value or the result of an expression to a variable.

B

Break

This statement is used to exit the loop.

Built-in Functions

They are ready to use as they are already coded.

C

Canvas

It is an area on a Web page that acts as a container for embedding graphic objects. It allows dynamic rendering of bitmap images and 2D shapes by using JavaScript.

Cascading Style Sheets (CSS)

It is a collection of styles that allows you to change the appearance of HTML elements on Web pages.

Class Selector

It is used to specify styles for a group of elements. It is used when there is a need to apply the same style on different types of elements in the HTML document.

Client-side scripting

It refers to scripts that are executed at the client-side by the Web browser, running on the user's computer.

Colorbox Plug-in

It enables displaying larger size images on the same Web page when a user clicks a hyperlink or an image.

Comments

These are statements that are not executed by the interpreter but are used to enhance the readability and understandability of the code.

Comparison Operator

It is used to compare two values and perform an action on the basis of the comparison.

Conditional Construct

It allows you to execute a selective statement or a block of statements based on the result of the expression being evaluated.

Continue

This statement is used to bring the control to the beginning of the loop.

D

Dynamic Web page

It is a Web page that responds to user actions or is dynamically created by a Web program on the basis of a user's request.

E

Event

It is an action that happens on a Web page, such as a mouse click or loading of a Web page.

Event Listener

It is an object that waits for an event to occur and performs certain actions corresponding to it.

F

Form

It is an interactive Web page that is used to accept the user input.

Frames

It provides a mechanism for positioning and displaying several Web pages in different sections of a single browser window.

Function

It is a self-contained block of statements that has a name.

G

Galleria Plug-in

It is the jQuery plug-in used to create elegant and professional-looking image galleries.

Graph

It is a way of representing relationships among a collection of items.

H

Hyper Text Markup Language (HTML)

It is a versatile markup language that can be used on different Web browsers to publish information as a Web page.

I

ID Selector

It is used to identify an element that you need to style differently from the rest of the page.

Images

It is used to add an artistic value to a Web page.

Image Gallery

It is a collection of thumbnail pictures or image links.

Image Rollover

It is an effect in which the appearance of an image changes when the mouse pointer is placed or moved on it.

J

JavaScript

It is used for client-side as well as server-side scripting.

jQuery

It is an open source and cross-browser library of JavaScript codes that is created to make the JavaScript programming simpler.

L

Logical Operator

It is used to evaluate complex expressions in which there is a need to evaluate a single or multiple expressions to assess the result.

Loop Structure

It is used to repeatedly execute one or more lines of

code.

M

Markup Language

It provides a way to describe the structure of a Web page, specifying how text or graphics are displayed on the Web page.

P

Path

It is a series of points joined together to create lines or shapes.

Preloading Images

It is a technique to load images in the browser cache before the script on the Web page is executed and the Web page is rendered in the browser window.

R

Regular Expression

It is a set of characters, which is used to specify a pattern.

Relative Path

It is the path of the file with respect to the current working directory.

RGraph

It is a JavaScript library that allows you to create different types of graphs on a Web page.

S

Selector

It is used to specify the HTML element that you want to style. You can have one or more selectors in a rule.

Server-side scripting

It refers to the scripts that are executed by the Web server on the basis of the user's request.

Static Web page

It is the web page that is delivered to the users exactly as these are stored. The content of these Web pages is not updated dynamically.

T

Tables

They are used for structuring and displaying complex information in a structured format on a Web page.

U

Uniform Resource Locator

It is the unique address of the website or a Web page on the Internet.

User-defined Functions

These are used to define your own functions according to your needs.

V

Variable

It is a named location in memory that is used to store a value.

W

World Wide Web (WWW)

It is a collection of resources on varied topics that can be accessed through the Internet.