Introduction: Web Fundamentals, HTML 5.0: basic tags, Form elements and attributes. Introduction to Cascading Style Sheets: CSS selectors, CSS BOX Model, CSS Positioning, and CSS floating. JQuery: Introduction to JavaScript, Selecting elements in the documents, Event handling, working with styles, The Event object, Using and creating plugins, JSON Fundamentals. Web-Based and REST Style Services:

Chapter-I

Web Fundamentals:

Constructing the software and data that provide all of this information requires knowledge of several different technologies, such as markup languages and meta-markup languages, as well as programming skills in a myriad of different programming languages, some specific to the World Wide Web and some designed for general-purpose computing.

The Internet is a collection of computers and other devices connected by equipment that allows them to communicate with each other. The Web is a collection of software and protocols that has been installed on most, if not all, of the computers on the Internet.

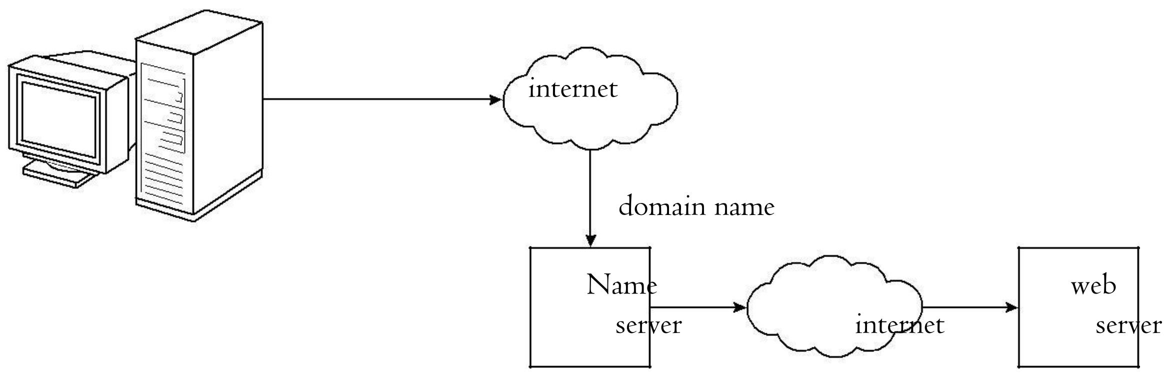
Internet: The Internet is a huge collection of computers connected in a communications network. These computers are of every imaginable size, configuration, and manufacturer. In fact, some of the devices connected to the Internet—such as plotters and printers—are not computers at all. The innovation that allows all of these diverse devices to communicate with each other is a single, low-level protocol: the Transmission Control Protocol/Internet Protocol (TCP/IP). TCP/IP became the standard for computer network connections in 1982. It can be used directly to allow a program on one computer to communicate with a program on another computer via the Internet. In most cases, however, a higher-level protocol runs on top of TCP/IP. Nevertheless, it's important to know that TCP/IP provides the low-level interface that allows most computers (and other devices) connected to the Internet to appear exactly the same.2 Rather than connecting every computer on the Internet directly to every other computer on the Internet, normally the individual computers in an organization are connected to each other in a local network. One node on this local network is physically connected to the Internet. So, the Internet is actually a network of networks, rather than a network of computers. Obviously, all devices connected to the Internet must be uniquely identifiable.

Internet Protocol Address: The Internet Protocol (IP) address of a machine connected to the Internet is a unique 32-bit number. IP addresses usually are written (and thought of) as four 8-bit numbers, separated by periods. The four parts are separately used by Internet-routing computers to decide where a message must go next to get to its destination. Organizations are assigned blocks of IPs, which they in turn assign to their machines that need Internet access—which now include most computers. For example, a small organization may be assigned 256 IP addresses, such as 191.57.126.0 to 191.57.126.255. Very large organizations, such as the Department of Defense, may be assigned 16 million IP addresses, which include IP addresses with one particular first 8-bit number, such as 12.0.0.0 to 12.255.255.255. Although people nearly always type domain names into their browsers, the IP works just as well. For example, the IP for United Airlines (www.ual.com) is 209.87.113.93. So, if a browser is pointed at http://209.87.113.93, it will be connected to the United Airlines Web site. In late 1998, a new IP standard, IPv6, was approved, although it still is not widely used. The most significant change was to expand the address size from 32 bits to 128 bits. This is a change that will soon be essential because the number of remaining unused IP addresses is diminishing rapidly.

Domain Names: people have difficulty dealing with and remembering numbers, machines on the Internet also have textual names. These names begin with the name of the host machine, followed by progressively larger enclosing collections of machines, called domains. There may be two, three, or more domain names. The first domain name, which appears immediately to the right of the host name, is the domain of which the host is a part. The second domain name gives the domain of which the first domain is a part. The last domain name identifies the type of organization in which the host resides, which is the largest domain in the site's name.

Example: For organizations in the United States, edu is the extension for educational institutions, com specifies a company, gov is used for the U.S. government, and org is used for many other kinds of organizations. In other countries, the largest domain is often an abbreviation for the country—for example, se is used for Sweden, and kz is used for Kazakhstan. Consider this sample address: movies.comedy.marxbros.com

Here, movies are the hostname and comedy is movies's local domain, which is a part of marxbros's domain, which is a part of the com domain. The hostname and all of the domain names are together called a fully qualified domain name. Because IP addresses are the addresses used internally by the Internet, the fully qualified domain name of the destination for a message, which is what is given by a browser user, must be converted to an IP address before the message can be transmitted over the Internet to the destination. These conversions are done by software systems called name servers, which implement the Domain Name System (DNS). Name servers serve a collection of machines on the Internet and are operated by organizations that are responsible for the part of the Internet to which those machines are connected. All document requests from browsers are routed to the nearest name server. If the name server can convert the fully qualified domain name to an IP address, it does so. If it cannot, the name server sends the fully qualified domain name to another name server for conversion. Like IP addresses, fully qualified domain names must be unique. Figure 1.1 shows how fully qualified domain names requested by a browser are translated into IPs before they are routed to the appropriate Web server.



__World Wide Web:__

Web Browsers: When two computers communicate over some network, in many cases one acts as a client and the other as a server. The client initiates the communication, which is often a request for information stored on the server, which then sends that information back to the client. The Web, as well as many other systems, operates in this client-server configuration. Documents provided by servers on the Web are requested by browsers, which are programs running on client machines. They are called browsers because they allow the user to browse the resources available on servers. The first browsers were text based—they were not capable of displaying graphic information, nor did they have a graphical user interface. This limitation effectively constrained the growth of the Web.

Web servers: Web servers are programs that provide documents to requesting browsers. Servers are slave programs: They act only when requests are made to them by browsers running on other computers on the Internet. The most commonly used Web servers are Apache, which has been implemented for a variety of computer platforms, and Microsoft's Internet Information Server (IIS), which runs under Windows operating systems.

Web server operations: Although having clients and servers is a natural consequence of information distribution, this configuration offers some additional benefits for the Web. On the one hand, serving information does not take a great deal of time. On the other hand, displaying information on client screens is time consuming. Because Web servers need not be involved in this display process, they can handle many clients. So, it is both a natural and an efficient division of labor to have a small number of servers provide documents to a large number of clients. Web browsers initiate network communications with servers by sending them URLs (discussed in Section 1.5). A URL can specify one of two different things: the address of a data file stored on the server that is to be sent to the client, or a program stored on the server that the client wants executed, with the output of the program returned to the client. All the communications between a Web client and a Web server use the standard Web protocol, Hypertext Transfer Protocol (HTTP).

General server Characteristics: The file structure of a Web server has two separate directories. The root of one of these is called the document root. The file hierarchy that grows from the document root stores the Web documents to which the server has direct access and normally serves to clients. The root of the other directory is called the server root. This directory, along with its descendant directories, stores the server and its support software. The files stored directly in the document root are those available to clients through top-level URLs. Typically, clients do not access the document root directly in URLs; rather, the server maps requested URLs to the document root, whose location is not known to clients. For example, suppose that the site name is www.tunias.com (not a real site—at least, not yet), which we will assume to be a UNIX-based system. Suppose further that the document root is named topdocs and is stored in the /admin/web directory, making its address /admin/web/topdocs. A request for a file from a client with the URL http://www.tunias.com/petunias.html will cause the server to search for the file with the file path /admin/web/topdocs/petunias.html. Likewise, the URL http://www.tunias.com/bulbs/tulips.html will cause the server to search for the file with the address /admin/web/topdocs/ bulbs/tulips.html. Many servers allow part of the servable document collection to be stored outside the directory at the document root. The secondary areas from which documents can be served are called virtual document trees.

Apache: Apache began as the NCSA server, httpd, with some added features. The name Apache has nothing to do with the Native American tribe of the same name. Rather, it came from the nature of its first version, which was a patchy version of the httpd server. As seen in the usage statistics given at the beginning of this section, Apache is the most widely used Web server. The primary reasons are as follows: Apache is an excellent server because it is both fast and reliable. Furthermore, it is open-source software, which means that it is free and is managed by a large team of volunteers, a process that efficiently and effectively maintains the system. Finally, it is one of the best available servers for Unix-based systems, which are the most popular for Web servers.

Apache is capable of providing a long list of services beyond the basic process of serving documents to clients. When Apache begins execution, it reads its configuration information from a file and sets its parameters to operate accordingly. A new copy of Apache includes default configuration information for a "typical" operation. The site manager modifies this configuration information to fit his or her particular needs and tastes.

IIS: Apache has been ported to the Windows platforms, it is not the most popular server on those systems. Because the Microsoft IIS server is supplied as part of Windows—and because it is a reasonably good server—most Windowsbased Web servers use IIS. Apache and IIS provide similar varieties of services. From the point of view of the site manager, the most important difference between Apache and IIS is that Apache is controlled by a configuration file that is edited by the manager to change Apache's behavior. With IIS, server behavior is modified by changes made through a window-based management program, named the IIS snap-in, which controls both IIS and ftp. This program allows the site manager to set parameters for the server. Under Windows XP and Vista, the IIS snap-in is accessed by going to Control Panel, Administrative Tools, and IIS Admin. Clicking on this last selection takes you to a window that allows starting, stopping, or pausing IIS. This same window allows IIS parameters to be changed when the server has been stopped.

Chapter-2
HTML Basics

Welcome to HTML Basics. This workshop leads you through the basics of Hyper Text Markup Language (HTML). HTML is the building block for web pages. You will learn to use HTML to author an HTML page to display in a web browser.

Objectives:

By the end of this workshop, you will be able to:

- Use a text editor to author an HTML document.
- Be able to use basic tags to denote paragraphs, emphasis or special type.
- Create hyperlinks to other documents.
- Create an email link.
- Add images to your document.
- Use a table for layout.
- Apply colors to your HTML document.

What is an html File?

HTML is a format that tells a computer how to display a web page. The documents themselves are plain text files with special "tags" or codes that a web browser uses to interpret and display information on your computer screen.

- HTML stands for Hyper Text Markup Language
- An HTML file is a text file containing small markup tags
- The markup tags tell the Web browser how to display the page
- An HTML file must have an htm or html file extension

Open your text editor and type the following text:

```
<html>
<head>
<title>My First Webpage</title>
</head>
<body>
This is my first homepage. <b>This text is bold</b> </body>
</html>
```

Save the file as **mypage.html**. Start your Internet browser. Select **Open** (or Open Page) in the **File** menu of your browser. A dialog box will appear. Select **Browse** (or Choose File) and locate the html file you just created - **mypage.html** - select it and click **Open**. Now you should see an address in the dialog box, for example **C:\MyDocuments\mypage.html**. Click **OK**, and the browser will display the page. To view how the page should look, visit this web page: **http://profdevtrain.austincc.edu/html/mypage.html**

Example Explained

What you just made is a skeleton html document. This is the minimum required information for a web document and all web documents should contain these basic components. The first tag in your html document is <html>. This tag tells your browser that this is the start of an html document. The last tag in your document is </html>. This tag tells your browser that this is the end of the html document.

- The text between the <head> tag and the </head> tag is header information. Header information is not displayed in the browser window.
- The text between the <title> tags is the title of your document. The <title> tag is used to uniquely identify each document and is also displayed in the title bar of the browser window.
- The text between the <body> tags is the text that will be displayed in your browser.
- The text between the <b> and </b> tags will be displayed in a bold font.

HTM or HTML Extension?

When you save an HTML file, you can use either the .htm or the .html extension. The .htm extension comes from the past when some of the commonly used software only allowed three letter extensions. It is perfectly safe to use either .html or .htm, but be consistent. **mypage.htm** and mypage.html are treated as different files by the browser.

How to View HTML Source

A good way to learn HTML is to look at how other people have coded their html pages. To find out, simply click on the View option in your browsers toolbar and select Source or Page Source. This will open a window that shows you the actual HTML of the page. Go ahead and view the source html for this page.

HTML Tags

What are HTML tags?

- HTML tags are used to mark-up HTML elements
- HTML tags are surrounded by the two characters < and >

- The surrounding characters are called angle brackets
- HTML tags normally come in pairs like <b> and </b>
- The first tag in a pair is the start tag, the second tag is the end tag
- The text between the start and end tags is the element content
- HTML tags are not case sensitive, <b> means the same as <B>

Logical vs. Physical Tags

In HTML there are both logical tags and physical tags. Logical tags are designed to describe (to the browser) the enclosed text's meaning. An example of a logical tag is the <strong> </strong> tag. By placing text in between these tags you are telling the browser that the text has some greater importance. By default all browsers make the text appear bold when in between the <strong> and
</strong> tags.

Physical tags on the other hand provide specific instructions on how to display the text they enclose. Examples of physical tags include:

- <b>: Makes the text bold.
- <big>: Makes the text usually one size bigger than what's around it.
- <i>: Makes text italic.

Physical tags were invented to add style to HTML pages because style sheets were not around, though the original intention of HTML was to not have physical tags. Rather than use physical tags to style your HTML pages, you should use style sheets.

HTML Elements

Remember the HTML example from the previous page:

```
<html>
<head>
<title>My First Webpage</title>
</head>
<body>
This is my first homepage. <b>This text is bold</b> </body>
</html>
```

This is an HTML element:

<b>**This text is bold**</b>
The HTML element begins with a start tag: <b>
The content of the HTML element is: This text is bold
The HTML element ends with an end tag: </b>
The purpose of the <b> tag is to define an HTML element that should be displayed as bold.

This is also an HTML element:
<body>
This is my first homepage. <b>**This text is bold**</b> </body>

This HTML element starts with the start tag <body>, and ends with the end tag </body>. The purpose of the <body> tag is to define the HTML element that contains the body of the HTML document.

Nested Tags

You may have noticed in the example above, the <body> tag also contains other tags, like the <b> tab. When you enclose an element in with multiple tags, the last tag opened should be the first tag closed. For example:

<p><b><em>***This is NOT the proper way to close nested tags.***</p></em></b>

<p><b><em>*This is the proper way to close nested tags.*</em></b></p>

Note: It doesn't matter which tag is first, but they must be closed in the proper order.

Why Use Lowercase Tags?

You may notice we've used lowercase tags even though I said that HTML tags are not case sensitive. <B> means the same as <b>. The World Wide Web Consortium (W3C), the group responsible for developing web standards, recommends lowercase tags in their HTML 4 recommendation, and XHTML (the next generation HTML) requires lowercase tags.

Tag Attributes

Tags can have attributes. Attributes can provide additional information about the HTML elements on your page. The <tag> tells the browser to do something, while the attribute tells the browser how to do it. For instance, if we add the bgcolor attribute, we can tell the browser that the background color of your page should be blue, like this: <body bgcolor="blue">.

This tag defines an HTML table: <table>. With an added border attribute, you can tell the browser that the table should have no borders: <table border="0">. Attributes always come in name/value pairs like this: name="value". Attributes are always added to the start tag of an HTML element and the value is surrounded by quotes.

Quote Styles, "red" or 'red'?

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed. In some rare situations, like when the attribute value itself contains quotes, it is necessary to use single quotes:

name='George "machine Gun" Kelly'

Note: Some tags we will discuss are deprecated, meaning the World Wide Web Consortium (W3C) the governing body that sets HTML, XML, CSS, and other technical standards decided those tags and attributes are marked for deletion in future versions of HTML and XHTML.Browsers should continue to support deprecated tags and attributes, but eventually these tags are likely to become obsolete and so future support cannot be guaranteed.

For a complete list of tags, visit W3C.org.

Basic HTML Tags

The most important tags in HTML are tags that define headings, paragraphs and line breaks.

Basic HTML Tags

Tag	Description
<html>	Defines an HTML document
<body>	Defines the document's body
<h1> to <h6>	Defines header 1 to header 6
<p>	Defines a paragraph
<br>	Inserts a single line break
<hr>	Defines a horizontal rule
<!-->	Defines a comment

Headings

Headings are defined with the <h1> to <h6> tags. <h1> defines the largest heading while <h6> defines the smallest.

<h1>This is a heading</h1>

<h2>This is a heading</h2>

<h3>This is a heading</h3>

<h4>This is a heading</h4>

<h5>This is a heading</h5>

<h6> This is a heading</h6>

HTML automatically adds an extra blank line before and after a heading. A useful heading attribute is align.

<h5 align="left">I can align headings </h5>

<h5 align="center">This is a centered heading </h5>

<h5 align="right">This is a heading aligned to the right </h5>

Paragraphs

Paragraphs are defined with the <p> tag. Think of a paragraph as a block of text. You can use the align attribute with a paragraph tag as well.

<p align="left">This is a paragraph</p>

<p align="center">this is another paragraph</p>

Important: You must indicate paragraphs with <p> elements. A browser ignores any indentations or blank lines in the source text. Without <p> elements, the document becomes one large paragraph. HTML automatically adds an extra blank line before and after a paragraph.

Line Breaks

The <br> tag is used when you want to start a new line, but don't want to start a new paragraph. The <br> tag forces a line break wherever you place it. It is similar to single spacing in a document.

This Code	Would Display
<p>This <br> is a para<br> graph with line breaks</p>	This<br>is a para<br>graph with line breaks

The <br> tag has no closing tag.

Horizontal Rule

The <hr> element is used for horizontal rules that act as dividers between sections, like this:

The horizontal rule does not have a closing tag. It takes attributes such as align and width. For instance:

This Code	Would Display
<hr width="50%" align="center">	———————————

Comments in HTML

The comment tag is used to insert a comment in the HTML source code. A comment can be placed anywhere in the document and the browser will ignore everything inside the brackets. You can use comments to write notes to yourself, or write a helpful message to someone looking at your source code.

This Code	Would Display
<p> This html comment would <!-- This is a comment --> be displayed like this.</p>	This HTML comment would be displayed like this.

Notice you don't see the text between the tags <!-- and -->. If you look at the source code, you would see the comment. To view the source code for this page, in your browser window, select **View** and then select **Source**.

Note: You need an exclamation point after the opening bracket <!-- but not before the closing bracket -->.

HTML automatically adds an extra blank line before and after some elements, like before and after a paragraph, and before and after a heading. If you want to insert blank lines into your document, use the <br> tag.

Try It Out!

Open your text editor and type the following text:

```
<html>
<head>
<title>My First Webpage</title>
</head>
<body>
<h1 align="center">My First Webpage</h1>
<p>Welcome to my first web page. I am writing this page using a text editor and plain old html.</p>
<p>By learning html, I'll be able to create web pages like a pro....<br> which I am of course.</p>
</body>
</html>
```

Save the page as **mypage2.html**. Open the file in your Internet browser.

Other HTML Tags

As mentioned before, there are logical styles that describe what the text should be and physical styles which actually provide physical formatting. It is recommended to use the logical tags and use style sheets to style the text in those tags.

.Logical Tags .

Tag	Description
<abbr>	Defines an abbreviation
<acronym>	Defines an acronym
<address>	Defines an address element
<cite>	Defines a *citation*
<code>	Defines computer code text
<blockquote>	Defines a long quotation
<del>	Defines text
<dfn>	Defines a *definition* term
<em>	Defines *emphasized* text
<ins>	Defines inserted text
<kbd>	Defines keyboard text
<pre>	Defines preformatted text

<q>	Defines a short quotation
<samp>	Defines sample computer code
<strong>	Defines **strong** text
<var>	Defines a *variable*

Physical Tags

Tag	Description
<b>	Defines **bold** text
<big>	Defines big text
<i>	Defines *italic* text
<small>	Defines small text
<sup>	Defines superscripted text
<sub>	Defines subscripted text
<tt>	Defines teletype text
<u>	Deprecated. Use styles instead

Character tags like <strong> and <em> produce the same physical display as <b> and <i> but are more uniformly supported across different browsers.

Some Examples:

The following paragraph uses the <blockquote> tag. In the previous sentence, the blockquote tag is enclosed in the <samp> Sample tag.

> We the people of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our Posterity, do ordain and establish this Constitution for the United States of America.

Although most browsers render blockquoted text by indenting it, that's not specifically what it's designed to do. It's conceivable that some future browser may render blockquoted text in some other way. However, for the time being, it is perfectly safe to indent blocks of text with the <blockquote>.

This Code	Would Display
<abbr title="World Wide Web">WWW</abbr>	WWW

When you hold your mouse pointer over the WWW, text in the title attribute will appear in.

HTML Character Entities

Some characters have a special meaning in HTML, like the less than sign (<) that defines the start of an HTML tag. If we want the browser to actually display these characters we must insert character entities in place of the actual characters themselves.

The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	quotation mark	&quot;	&#34;

'	apostrophe	&apos; (does not work in IE)	&#39;

A character entity has three parts: an ampersand (&), an entity name or an entity number, and finally a semicolon (;). The & means we are beginning a special character, the **;** means ending a special character and the letters in between are sort of an abbreviation for what it's for. To display a less than sign in an HTML document we must write: **&lt;** or **&#60;** The advantage of using a name instead of a number is that a name is easier to remember. The disadvantage is that not all browsers support the newest entity names, while the support for entity numbers is very good in almost all browsers.

Note: Entities are case sensitive.

Non-breaking Space

The most common character entity in HTML is the non-breaking space **&nbsp;**. Normally HTML will truncate spaces in your text. If you add 10 spaces in your text, HTML will remove 9 of them. To add spaces to your text, use the &nbsp; character entity.

This Code	Would Display
<p> This code would appear as this.</p>	This code would appear as this.

This Code	Would Display
<p> This code &nbsp;&nbsp;&nbsp; would appear with three extra spaces.</p>	This code would appear with three extra spaces.

HTML Fonts

The <font> tag in HTML is deprecated. The World Wide Web Consortium (W3C) has removed the <font> tag from its recommendations. In future versions of HTML, style sheets (CSS) will be used to define the layout and display properties of HTML elements.

The <font> Tag Should **NOT** be used.

HTML Backgrounds

Backgrounds

The <body> tag has two attributes where you can specify backgrounds. The background can be a color or an image.

Bgcolor

The bgcolor attribute specifies a background-color for an HTML page. The value of this attribute can be a hexadecimal number, an RGB value, or a color name:

```
<body bgcolor="#000000">
<body bgcolor="rgb(0,0,0)">
<body bgcolor="black">
```

The lines above all set the background-color to black.

Background

The background attribute can also specify a background-image for an HTML page. The value of this attribute is the URL of the image you want to use. If the image is smaller than the browser window, the image will repeat itself until it fills the entire browser window.

```
<body background="clouds.gif">
```

```
<body background="http://profdevtrain.austincc.edu/html/graphics/clouds.gif">
```

The URL can be relative (as in the first line above) or absolute (as in the second line above). If you want to use a background image, you should keep in mind:

- Will the background image increase the loading time too much?
- Will the background image look good with other images on the page?
- Will the background image look good with the text colors on the page?
- Will the background image look good when it is repeated on the page?
- Will the background image take away the focus from the text?

Note: The bgcolor, background, and the text attributes in the <body> tag are deprecated in the latest versions of HTML (HTML 4 and XHTML). The World Wide Web Consortium (W3C) has removed these attributes from its recommendations. Style sheets (CSS) should be used instead (to define the layout and display properties of HTML elements).

Open your text editor and type the following text:

```
<html>
<head>
<title>My First Webpage</title>
</head>
<body background="http://profdevtrain.austincc.edu/html/graphics/clouds.gif"
bgcolor="#EDDD9E">
<h1 align="center">My First Webpage</h1>
<p>Welcome to my <strong>first</strong> webpage. I am writing this page using a text editor and plain old
html.</p>
<p>By learning html, I'll be able to create webpages like a <del>beginner</del> pro....<br>
which I am of course.</p>
</body>
</html>
```

Save your page as **mypage3.html** and view it in your browser.
Notice we gave our page a background color as well as a background image. If for some reason the web page is unable to find the picture, it will display our background color.

HTML Colors

Color Values

Colors are defined using a hexadecimal notation for the combination of red, green, and blue color values (RGB). The lowest value that can be given to one light source is 0 (hex #00). The highest value is 255 (hex #FF). This table shows the result of combining red, green, and blue:

Color	Color HEX	Color RGB
⬛	#000000	rgb(0,0,0)
🟥	#FF0000	rgb(255,0,0)
🟩	#00FF00	rgb(0,255,0)
🟦	#0000FF	rgb(0,0,255)
🟨	#FFFF00	rgb(255,255,0)
▨	#00FFFF	rgb(0,255,255)
🟪	#FF00FF	rgb(255,0,255)
▨	#C0C0C0	rgb(192,192,192)
⬜	#FFFFFF	rgb(255,255,255)

Color Names

A collection of color names is supported by most browsers. To view a table of color names that are supported by most browsers visit this web page: **http://profdevtrain.austincc.edu/html/color_names.htm**

> **Note:** Only 16 **color names** are supported by the W3C HTML 4.0 standard (aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow). For all other colors you should use the **Color HEX** value.

Color	Color HEX	Color Name
⬜	#F0F8FF	AliceBlue
⬜	#FAEBD7	AntiqueWhite
🟩	#7FFFD4	Aquamarine
⬛	#000000	Black
🟦	#0000FF	Blue
🟪	#8A2BE2	BlueViolet
🟫	#A52A2A	Brown

Web Safe Colors

A few years ago, when most computers supported only 256 different colors, a list of 216 Web Safe Colors was suggested as a Web standard. The reason for this was that the Microsoft and Mac operating system used 40 different "reserved" fixed system colors (about 20 each). This 216 cross platform web safe color palette was originally created to ensure that all computers would display all colors correctly when running a 256 color palette.

16 Million Different Colors

The combination of Red, Green and Blue values from 0 to 255 gives a total of more than 16 million different colors to play with (256 x 256 x 256). Most modern monitors are capable of displaying at least 16,384 different colors.

HTML Lists

HTML provides a simple way to show unordered lists (bullet lists) or ordered lists (numbered lists).

Unordered Lists

An unordered list is a list of items marked with bullets (typically small black circles). An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

This Code	Would Display
`<ul>`	
`<li>Coffee</li>`	• Coffee
`<li>Milk</li>`	• Milk
`</ul>`	

Ordered Lists

An ordered list is also a list of items. The list items are marked with numbers. An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

This Code	Would Display
`<ol>`	
`<li>Coffee</li>`	1. Coffee
`<li>Milk</li>`	2. Milk
`</ol>`	

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Definition Lists

Definition lists consist of two parts: a **term** and a **description**. To mark up a definition list, you need three HTML elements; a container <dl>, a definition term <dt>, and a definition description <dd>.

This Code	Would Display
<dl> <dt>Cascading Style Sheets</dt> <dd>Style sheets are used to provide presentational suggestions for documents marked up in HTML. </dd> </dl>	Cascading Style Sheets Style sheets are used to provide presentational suggestions for documents marked up in HTML.

Inside a definition-list definition (the <dd> tag) you can put paragraphs, line breaks, images, links, other lists, etc

Try It Out

Open your text editor and type the following:

```
<html>
<head>
<title>My First Webpage</title>
</head>
<body bgcolor="#EDDD9E">
<h1 align="center">My First Webpage</h1>
<p>Welcome to my <strong>first</strong> webpage. I am writing this page using a text editor and plain old
html.</p>
<p>By learning html, I'll be able to create web pages like a pro....<br> which I am of
course.</p>
Here's what I've learned: <ul>
<li>How to use HTML tags</li>
<li>How to use HTML colors</li>
<li>How to create Lists</li> </ul>
</body>
</html>
```

HTML Links

HTML uses the <a> anchor tag to create a link to another document or web page.

The Anchor Tag and the Href Attribute

An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc. The syntax of creating an anchor:

```
<a href="url">Text to be displayed</a>
```

The <a> tag is used to create an anchor to link from, the href attribute is used to tell the address of the document or page we are linking to, and the words between the open and close of the anchor tag will be displayed as a hyperlink.

This Code	Would Display

<a href="http://www.austincc.edu/">Visit ACC!</a>	Visit ACC!

The Target Attribute

With the target attribute, you can define **where** the linked document will be opened. By default, the link will open in the current window. The code below will open the document in a new browser window:

<a href=http://www.austincc.edu/ target="_blank">Visit ACC!</a>

Email Links

To create an email link, you will use mailto: plus your email address. Here is a link to ACC's Help Desk:

<a href="mailto:helpdesk@austincc.edu">Email Help Desk</a>

To add a subject for the email message, you would add ?subject= after the email address. For example:

<a href="mailto:helpdesk@austincc.edu?subject=Email Assistance">Email Help Desk</a>

The Anchor Tag and the Name Attribute

The name attribute is used to create a named anchor. When using named anchors we can create links that can jump directly to a specific section on a page, instead of letting the user scroll around to find what he/she is looking for. Unlike an anchor that uses href, a named anchor doesn't change the appearance of the text (unless you set styles for that anchor) or indicate in any way that there is anything special about the text. Below is the syntax of a named anchor:

<a name="top">Text to be displayed</a>

To link directly to the top section, add a # sign and the name of the anchor to the end of a URL, like this:

This Code	Would Display
<a href="http://profdevtrain.austincc.edu/html/10links.html#top">Back to top of page </a>	Back to top of page
A hyperlink to the top of the page from within the file 10links.html will look like this: <a href="#top">Back to top of page </a>	Back to top of page

Note: Always add a trailing slash to subfolder references. If you link like this: href="http://profdevtrain.austincc.edu/html", you will generate two HTTP requests to the server, because the server will add a slash to the address and create a new request like this: href="http://profdevtrain.austincc.edu/html/"

Named anchors are often used to create "table of contents" at the beginning of a large document. Each chapter within the document is given a named anchor, and links to each of these anchors are put at the top of the document. If a browser cannot find a named anchor that has been specified, it goes to the top of the document. No error occurs.

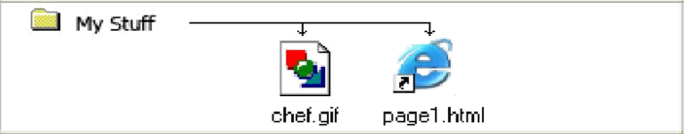
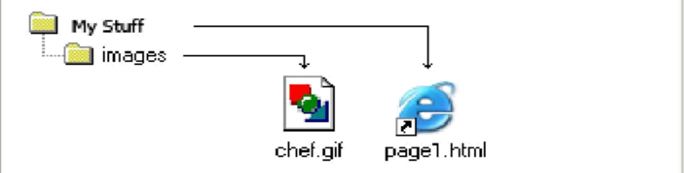
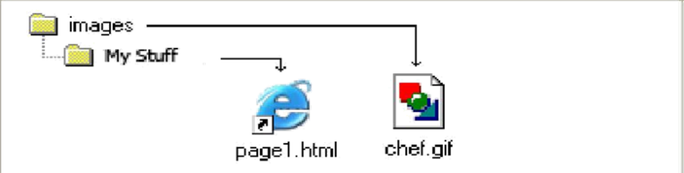
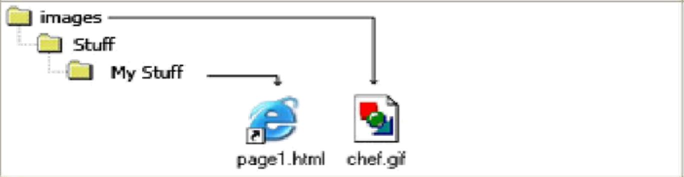
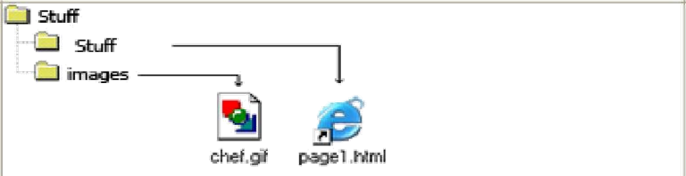
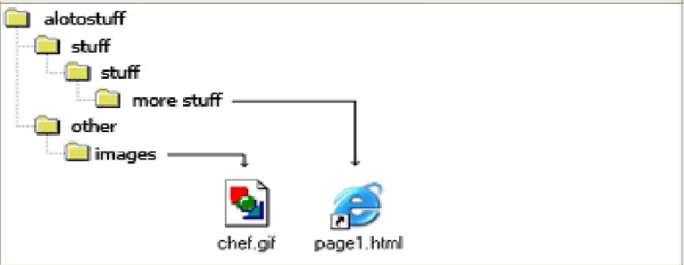
HTML Images

The Image Tag and the Src Attribute

The <img> tag is empty, which means that it contains attributes only and it has no closing tag. To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display on your page. The syntax of defining an image:

This Code	Would Display
<img src="graphics/chef.gif">	

Not only does the source attribute specify what image to use, but where the image is located. The above image, graphics/chef.gif, means that the browser will look for the image name **chef.gif** in a **graphics** folder in the same folder as the html document itself.

	src="chef.gif" means that the image is in the same folder as the html document calling for it.
	src="images/chef.gif" means that the image is one folder down from the html document that called for it. This can go on down as many layers as necessary.
	src="images/chef.gif" means that the image is one folder down from the html document that called for it. This can go on down as many layers as necessary.
	src="../chef.gif" means that the image is in one folder up from the html document that called for it.
	src="../../chef.gif" means that the image is two folders up from the html document that called for it.
 alotostuff stuff stuff more stuff other images chef.gif page1.html	src="../images/chef.gif" means that the image is one folder up and then another folder down in the images directory. src="../../../other/images/chef.gif" means this goes multiple layers up.

The browser puts the image where the image tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

The Alt Attribute

The alt attribute is used to define an alternate text for an image. The value of the alt attribute is author-defined text:

```
<img src="graphics/chef.gif" alt="Smiling Happy Chef ">
```

The alt attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image. It is a good practice to include the alt attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers or use screen readers.

Image Dimensions

When you have an image, the browser usually figures out how big the image is all by itself. If you put in the image dimensions in pixels however, the browser simply reserves a space for the image, then loads the rest of the page. Once the entire page is loads it can go back and fill in the images. Without dimensions, when it runs into an image, the browser has to pause loading the page, load the image, then continue loading the page. The chef image would then be:

```
<img src="graphics/chef.gif" width="130" height="101" alt="Smiling Happy Chef">
```

Open the file **mypage2.html** in your text editor and add code highlighted in bold:

```
<html>
<head>
<title>My First
Webpage</title> </head>
<body>
<h1 align="center">My First Web page</h1>
<p>Welcome to my first webpage. I am writing this page using a text editor and plain old html.</p>
<p>By learning html, I'll be able to create web pages like a pro....<br>
which I am of course.</p>
<!-- Who would have guessed how easy this would be :) -->
<p><img src="graphics/chef.gif" width="130" height="101" alt="Smiling Happy Chef"
align="center"></p>
<p align="center">This is my Chef</p>
</body>
</html>
```

Tables

Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). The letters td stands for table data, which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

This Code	Would Display
`<table>`	
`<tr>`	
`<td>row 1, cell 1</td>`	
`<td>row 1, cell 2</td>`	
`</tr>`	row 1, cell 1 row 1, cell 2
`<tr>`	row 2, cell 1 row 2, cell 2

`<td>row 2, cell 1</td>`
`<td>row 2, cell 2</td>`
`</tr>`
`</table>`	

Tables and the Border Attribute

To display a table with borders, you will use the border attribute.

This Code	Would Display

`<table border="1">`
`<tr>`
 	
`<td>Row 1, cell 1</td>`
`<td>Row 1, cell 2</td>`
`</tr>`
`</table>`
	row 1, cell 1 row 1, cell 2

This Code	Would Display

`<table border="5">`
`<tr>`
 	
`<td>Row 1, cell 1</td>`
`<td>Row 1, cell 2</td>`
 	
`</tr>`
`</table>`
	row 1, cell 1 row 1, cell 2

Open up your text editor. Type in your <html>, <head> and <body> tags. From here on I will only be writing what goes between the <body> tags. Type in the following:

```
<table
border="1"> <tr>
<td>Tables can be used to layout information</td>
<td>&nbsp; <img src="http://profdevtrain.austincc.edu/html/graphics/chef.gif"> &nbsp; </td>
</tr>
</table>
```

Headings in a Table

Headings in a table are defined with the <th> tag.

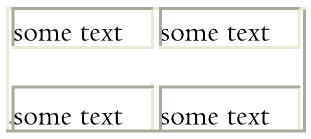
This code	Would Display

`<table border="1">`
`<tr>`
`<th>Heading</th>`
`<th>Another Heading</th>`
`</tr>`
`<tr>`
 	
`<td>row 1, cell 1</td>`
`<td>row 1, cell 2</td>`
`</tr>`
`<tr>`
`<td>row 2, cell 1</td>`
`<td>row 2, cell 2</td>`
	Heading Another Heading
	row 1, cell 1 row 1, cell 2
	row 2, cell 1 row 2, cell 2

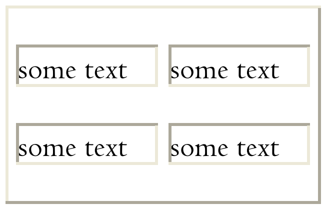
`</tr>` `</table>`	

Cell Padding and Spacing

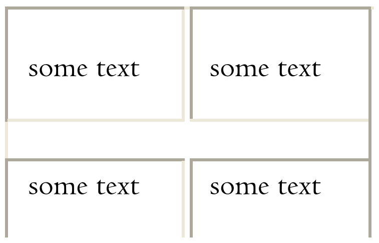
The `<table>` tag has two attributes known as cellspacing and cellpadding. Here is a table example without these properties. These properties may be used separately or together.

This Code	Would Display
`<table border="1">` `<tr>` `<td>some text</td>` `<td>some text</td>` `</tr>` `<tr>` `<td>some text</td>` `<td>some text</td>` `</tr>` `</table>`	 some text some text some text some text

Cellspacing is the pixel width between the individual data cells in the table (The thickness of the lines making the table grid). The default is zero. If the border is set at 0, the cellspacing lines will be invisible.

This Code	Would Display
`<table border="1" cellspacing="5">` `<tr>` `<td>some text</td>` `<td>some text</td>` `</tr><tr>` `<td>some text</td>` `<td>some text</td>` `</tr>` `</table>`	 some text some text some text some text

Cellpadding is the pixel space between the cell contents and the cell border. The default for this property is also zero. This feature is not used often, but sometimes comes in handy when you have your borders turned on and you want the contents to be away from the border a bit for easy viewing. Cellpadding is invisible, even with the border property turned on. Cellpadding can be handled in a style sheet.

This Code	Would Display
`<table border="1" cellpadding="10">` `<tr>` `<td>some text</td>` `<td>some text</td>` `</tr><tr>` `<td>some text</td>` `<td>some text</td>`	 some text some text some text some text

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

Table Tags

Tag	Description
`<table>`	Defines a table
`<th>`	Defines a table header
`<tr>`	Defines a table row
`<td>`	Defines a table cell
`<caption>`	Defines a table caption
`<colgroup>`	Defines groups of table columns
`<col>`	Defines the attribute values for one or more columns in a table

Table Size

Table Width

The width attribute can be used to define the width of your table. It can be defined as a fixed width or a relative width. A fixed table width is one where the width of the table is specified in pixels. For example, this code, <table width="550">, will produce a table that is 550 pixels wide. A relative table width is specified as a percentage of the width of the visitor's viewing window. Hence this code, <table width="80%">, will produce a table that occupies 80 percent of the screen.

This table width is 250 pixels

This table width is 50%

There are arguments in favor of giving your tables a relative width because such table widths yield pages that work regardless of the visitor's screen resolution. For example, a table width of 100% will always span the entire width of the browser window whether the visitor has a 800x600 display or a 1024x768 display (etc). Your visitor never needs to scroll horizontally to read your page, something that is regarded by most people as being very annoying.

HTML Layout - Using Tables

One very common practice with HTML, is to use HTML tables to format the layout of an HTML age.
A part of this page is formatted with two columns. As you can see on this page, there is a left column and a right column. This text is displayed in the left column.
An HTML <table> is used to divide a part of this Web page into two columns. The trick is to use a table without borders, and maybe a little extra cell-padding. No matter how much text you add to this page, it will stay inside its column borders.

Let's put everything you've learned together to create a simple page. Open your text editor and type the following text:

```
<html>
<head>
<title>My First Web Page </title>
</head>
<body>
<table width="90%" cellpadding="5" cellspacing="0" > <tr
   bgcolor="#EDDD9E">
      <td width="200" valign="top"><img src="graphics/contact.gif" width="100"
```

```
height="100"></td>
        <td valign="top"><h1 align="right">Janet Doeson</h1>
        <h3 align="right">Technical Specialist</h3></td>
    </tr>
    <tr>
        <td
            width="200"
            >
            <h3>Menu<
            /h3> <ul>
                <li><a
                href="home.html">Home</a></li> <li>
                <a href="faq.html">FAQ</a></li>
                <li> <a href="contact.html">Contact</a></li>
                <li> <a href="http://www.austincc.edu">Links</a> </li>
        </ul></td>
        <td valign="top"><h2 align="center">Welcome!</h2>
        <p>Welcome to my first webpage. I created this webpage without the assistance of a webpage editor. Just
my little text editor and a keen understanding of html.</p>
        <p>Look around. Notice I'm able to use paragraphs, lists and headings. You may not be able to tell, but
the layout is done with a table. I'm very clever. </p>
        <blockquote>
            <p>I always wanted to be somebody, but now I realize I should have been more
specific.</p>                                           </td
            <cite>Lily Tomlin </cite> </blockquote>      >
    </tr>
</table>
<hr width="90%" align="left">
<address>
      Janet Doeson<br>
      Technical Specialist<br>
      512.555.5555
</address>
<p>Contact me at <a href="mailto:jdoeson@acme.com">jdoeson@acme.com</a> </p>

</body>
</html>
```

I have indented some of the HTML code in the above example. Indenting the code can make your HTML document easier to read.

Create Your Own Page

It's time to create your own page. Use your text editor to create a page which contains the following:

- the required HTML page codes
- link to another web page
- an email link
- a picture/graphic
- a list of information

HTML Forms

HTML Forms are required when you want to collect some data from the site visitor. For example during user registration you would like to collect information such as name, email address, credit card, etc.
A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.
There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.
The HTML **<form>** tag is used to create an HTML form and it has following syntax:

```
<form action="Script URL" method="GET|POST">
   form elements like input, textarea etc.
</form>
```

Form Attributes

Apart from common attributes, following is a list of the most frequently used form attributes:

Attribute	Description
action	Backend script ready to process your passed data.
method	Method to be used to upload data. The most frequently used are GET and POST methods.
target	Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
enctype	You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are: • **application/x-www-form-urlencoded** - This is the standard method most forms use in simple scenarios. • **mutlipart/form-data** - This is used when you want to upload binary data in the form of files like image, word file etc.

HTML Form Controls

There are different types of form controls that you can use to collect data using HTML form:

- Text Input Controls
- Checkboxes Controls
- Radio Box Controls
- Select Box Controls
- File Select boxes
- Hidden Controls
- Clickable Buttons
- Submit and Reset Button

Text Input Controls

There are three types of text input used on forms:

- **Single-line text input controls -** This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML **<input>** tag.
- **Password input controls -** This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tag.
- **Multi-line text input controls -** This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML **<textarea>** tag.

Single-line text input controls

This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.

Example

Here is a basic example of a single-line text input used to take first name and last name:

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Text Input Control</title>
</head>
<body>
<form >
First name:  <input type="text" name="first_name" />
<br>
Last name:  <input type="text" name="last_name" />
</form>
</body>
</html>
```

This will produce following result:

First name:

Last name:

Attributes

Following is the list of attributes for <input> tag for creating text field.

Attribute	Description
type	Indicates the type of input control and for text input control it will be set to **text**.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	This can be used to provide an initial value inside the control.
size	Allows to specify the width of the text-input control in terms of characters.
maxlength	Allows to specify the maximum number of characters a user can enter into the text box.

Password input controls

This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input> tag but type attribute is set to **password**.

Example

Here is a basic example of a single-line password input used to take user password:

```
<!DOCTYPE html>
<html>
<head>
<title>Password Input Control</title>
</head>
<body>
<form >
User ID :  <input type="text" name="user_id" />
<br>
Password:  <input type="password" name="password" />
</form>
</body>
</html>
```

This will produce following result:

User ID :

Password:

Attributes

Following is the list of attributes for <input> tag for creating password field.

Attribute	Description
type	Indicates the type of input control and for password input control it will be set to **password**.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.

value	This can be used to provide an initial value inside the control.
size	Allows to specify the width of the text-input control in terms of characters.
maxlength	Allows to specify the maximum number of characters a user can enter into the text box.

Multiple-Line Text Input Controls

This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

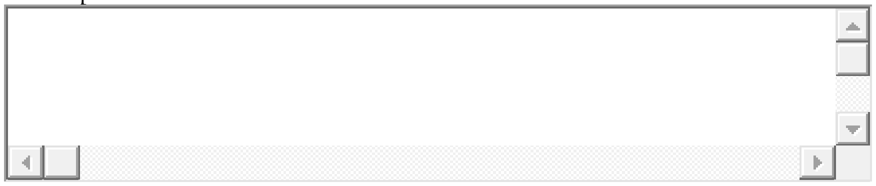
Example

Here is a basic example of a multi-line text input used to take item description:

```
<!DOCTYPE html>
<html>
<head>
<title>Multiple-Line Input Control</title>
</head>
<body>
<form>
Description : <br />
<textarea rows="5" cols="50" name="description">
Enter description here...
</textarea>
</form>
</body>
</html>
```

This will produce following result:

Description :



Attributes

Following is the list of attributes for <textarea> tag.

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
rows	Indicates the number of rows of text area box.
cols	Indicates the number of columns of text area box

Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **checkbox**.

Example

Here is an example HTML code for a form with two checkboxes:

```
<!DOCTYPE html>
<html>
<head>
<title>Checkbox Control</title>
</head>
<body>
<form>
<input type="checkbox" name="maths" value="on"> Maths
<input type="checkbox" name="physics" value="on"> Physics
</form>
```

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

This will produce following result:

☐ Maths ☐ Physics

Attributes

Following is the list of attributes for <checkbox> tag.

Attribute	Description
type	Indicates the type of input control and for checkbox input control it will be set to **checkbox**.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	The value that will be used if the checkbox is selected.
checked	Set to *checked* if you want to select it by default.

Radio Button Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **radio**.

Example

Here is example HTML code for a form with two radio buttons:

```
<!DOCTYPE html>
<html>
<head>
<title>Radio Box Control</title>
</head>
<body>
<form>
<input type="radio" name="subject" value="maths"> Maths
<input type="radio" name="subject" value="physics"> Physics
</form>
</body>
</html>
```

This will produce following result:

○ Maths ○ Physics

Attributes

Following is the list of attributes for radio button.

Attribute	Description
type	Indicates the type of input control and for checkbox input control it will be set to **radio**.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	The value that will be used if the radio box is selected.
checked	Set to *checked* if you want to select it by default.

Select Box Control

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

Example

Here is example HTML code for a form with one drop down box

```
<!DOCTYPE html>
<html>
<head>
<title>Select Box Control</title>
</head>
<body>
```

```
<form>
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
</form>
</body>
</html>
```

This will produce following result:

Attributes

Following is the list of important attributes of <select> tag:

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
size	This can be used to present a scrolling list box.
multiple	If set to "multiple" then allows a user to select multiple items from the menu.

Following is the list of important attributes of <option> tag:

Attribute	Description
value	The value that will be used if an option in the select box box is selected.
selected	Specifies that this option should be the initially selected value when the page loads.
label	An alternative way of labeling options

File Upload Box

If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to **file**.

Example

Here is example HTML code for a form with one file upload box:

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<input type="file" name="fileupload" accept="image/*" />
</form>
</body>
</html>
```

This will produce following result:

Attributes

Following is the list of important attributes of file upload box:

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
accept	Specifies the types of files that the server accepts.

Button Controls

There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input> tag by setting its type attribute to **button**. The type attribute can take the following values:

Type	Description

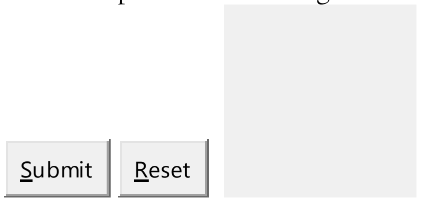
submit	This creates a button that automatically submits a form.
reset	This creates a button that automatically resets form controls to their initial values.
button	This creates a button that is used to trigger a client-side script when the user clicks that button.
image	This creates a clickable button but we can use an image as background of the button.

Example

Here is example HTML code for a form with three types of buttons:

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset"  value="Reset" />
<input type="button" name="ok" value="OK"  />
<input type="image" name="imagebutton" src="/html/images/logo.png" />
</form>
</body>
</html>
```

This will produce following result:



Hidden Form Controls

Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page has be displayed next based on the passed current page.

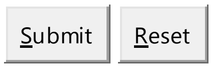
Example

Here is example HTML code to show the usage of hidden control:

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<p>This is page 10</p>
<input type="hidden" name="pagename" value="10" />
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset"  value="Reset" />
</form>
</body>
</html>
```

This will produce following result:

This is page 10

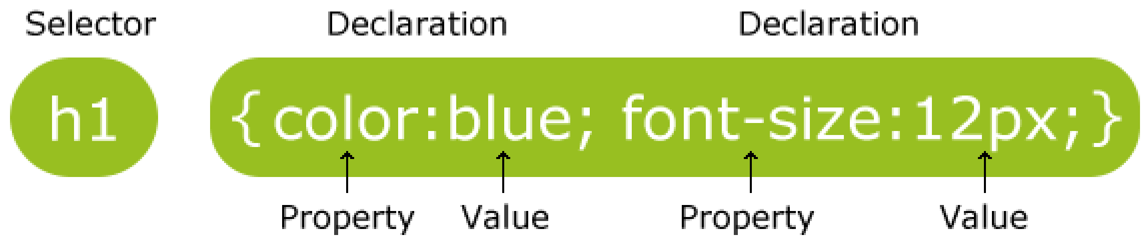


Syntactic differences between HTML and XHTML
Case sensitivity: In HTML, tag and attribute names are case insensitive meaning that ,, and
are equivalent. In XHTML, all tag and attribute names must be in lowercase.
Closing tags: In HTML, closing tag may be omitted if the processing agent can infer heir presence. For example, in HTML, paragraph elements often do not have closing tags. For example
<p>During Spring, flowers are born. …
<p>
During Fall, flowers die….. HTML documents are case insensitive whereas XHTML documents have to be in lowercase Closing tags may be omitted in HTML but not in XHTML where all elements must have closing tags except for content tags where it is not a must
<input type=text name=address/>
Quoted attribute values :all attribute values must be quoted whether it is numeric or character based
Explicit attribute values: In HTML some attributes are implicit Quoted attribute values: In HTML, attribute values must be quoted only if there are embedded special characters or white space characters.
Explicit attribute values: In HTML, some attribute values are implicit, that is, they need not be explicitly stated. For example, if the border attribute appears in a <table>tag without a value, it specifies a default width border on the table. For example: <table border> . id and name attributes. HTML markup often uses the name attribute for elements. Element nesting: Although HTML has rules against improper nesting of elements, they are not enforced.
Examples of nesting rules are:
1. An anchor element cannot contain another anchor element, and a form element cannot contain another form element.
2. If one element appears inside another element its closing tag should appear before the closing tag of the outer element.
3. Block elements cannot be nested inside inline elements.
4. Text cannot be nested directly in body or form elements.
5. List elements cannot be directly nested in list elements.

Chapeter-3
CSS

Introduction

The CSS1 specification was developed in 1996 by W3C. CSS2 was released in 1998 which added many properties and values to CSS1. CSS3 has been under development since the late 1990s. CSSs provide the means to control and change presentation of HTML documents. CSS is not technically HTML, but can be embedded in HTML documents. Cascading style sheets were introduced to provide a uniform and consistent way to specify presentation details in XHTML documents. Most of the style tags and attributes are those deprecated from HTML 4.0 in favor of style sheets. Idea of style sheets is not a new concept as it existed in desktop publishing systems and word processors. Style sheets allow you to impose a standard style on a whole document, or even a whole collection of documents. Style sheets can be defined at three levels to specify the style of a document .Hence called Cascading style sheets. Style is specified for a tag by the values of its properties.
A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

Levels of Style Sheets

There are three levels of style sheets, in order from lowest level to highest level, are inline, document level, and external. Inline style sheets are specified for a specific occurrence of a tag and apply only to the content of that tag. This application of style, which defeats the purpose of style sheets – that of imposing uniform style on the tags of at least one whole document. Another disadvantage of inline style sheets is that they result in style information, which is expressed in a expressed in a language distinct from XHTML markup, being embedded in various places in documents. Document-level style specifications appear in the document head section and apply to the whole body of the document. External style sheets are not part of the documents to which they apply. They are stored separately and are referenced in all documents that use them. They are written as text files with MIME type text/css.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
   color: red;
   text-align: center;
}
</style>
</head>
<body>

<p>Hello World!</p>
<p>These paragraphs are styled with CSS.</p>

</body>
</html>
```

Linking an External Style sheet A tag is used to specify that the browser is to fetch and use an external style sheet file through href. The href attribute of is used to specify the URL of the style sheet document, as in the following example: This link must appear in the head of the document. External style sheets can be validated, with the service http://jigsaw.w3.org/css-validator/ validator-upload.html. External style sheets can be added using other alternate style specification known as file import - @import url (filename); Filename is not quoted. Import appears only at the beginning of the content of a style element. The file imported can contain both markup as well as style rules.

Style specification formats:
The format of a style specification depends on the level of the style sheet. Inline style sheet appears as the value of the style attribute of the tag. The general form of which is as follows: style = "property_1: value_1; property_2: value_2; …
property_n: value_n;"
Format for Document-level Document style specifications appear as the content of a style element within the header of a document, although the format of the specification is quite different from that of inline style sheets. The <style> tag must include the type attribute, set to "text/css—(as there are other style sheets in JavaScript). The list of rules must be placed in a comment, because CSS is not XHTML. Style element must be placed within the header of a document. Comments in the rule list must have a different form use C comments (/*…*/). The general form of the content of a style element is as follows:
<style type= "text/css">
/* rule list(/* styles for paragraphs and other tags*/)
*/ </style>
Form of the rules:
Each style in a rule list has two parts: selector, which indicates the tag or tags affected by the rules. Each property/value pair has the form->property: value Selector { property_1: value_1; property_2: value_2:… property_n: value_n;} Pairs are separated by semicolons, just as in the value of a <style> tag

Selector forms:
Selector can have variety of forms like:
1. Simple selector form
2. Class selector
3. Generic selector
4. Id selector
5. Universal selector
6. Pseudo classes
Simple selector form Simple selector form is a list of style rules, as in the content of a <style> tag for document-level style sheets. The selector is a tag name or a list of tag names, separated by commas. Consider the following examples, in which the property is font-size and the property value is a number of points :
h1, h3 { font-size: 24pt ;}
h2 { font-size: 20pt ;}
Selectors can also specify that the style should apply only to elements in certain positions in the document .This is done by listing the element hierarchy in the selector.
• Contextual selectors: Selectors can also specify that the style should apply only to elements in certain positions in the document .
• In the eg selector applies its style to the content of emphasis elements that are descendants of bold elements in the body of the document. body b em {font-size: 24pt ;}
Class Selectors Used to allow different occurrences of the same tag to use different style specifications. A style class has a name, which is attached to the tag's name with a period.
p.narrow {property-value list} p.wide {property-value list} The class you want on a particular occurrence of a tag is specified with the class attribute of the tag.

Generic Selectors A generic class can be defined if you want a style to apply to more than one kind of tag. A generic class must be named, and the name must begin with a period without a tag name in its name.

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

Id Selectors An id selector allow the application of a style to one specific element. The general form of an id selector is as follows :

```
#specific-id {property-value list}
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```

Universal selector The universal selector, denoted by an asterisk(*), which applies style to all elements in the document.

```
*{color:red;}
```

Pseudo Classes

Pseudo classes are styles that apply when something happens, rather than because the target element simply exists. Names of pseudo classes begin with colons hover classes apply when the mouse cursor is over the element focus classes apply when an element has focus i.e. the mouse cursor is over the element and the left mouse button is clicked. These two pseudo classes are supported by FX2 but IE7 supports only hover.

```
selector:pseudo-class {
    property:value;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
   color: red;
}

/* visited link */
a:visited {
   color: green;
}

/* mouse over link */
a:hover {
   color: hotpink;
}

/* selected link */
a:active {
   color: blue;
}
</style>
</head>
<body>

<p><b><a href="default.html" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be
effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>

</body>
</html>
```

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

When you hover over the link in the example, it will change color:

```
<!DOCTYPE html>
<html>
<head>
<style>
a.highlight:hover {
   color: #ff0000;
}
</style>
</head>
<body>

<p><a class="highlight" href="css_syntax.html">CSS Syntax</a></p>
<p><a href="default.html">CSS Tutorial</a></p>
```

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

One more example on hover

Hover over a <div> element to show a <p> element (like a tooltip):

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
   display: none;
   background-color: yellow;
   padding: 20px;
}

div:hover p {
   display: block;
}
</style>
</head>
<body>

<div>Hover over me to show the p element
  <p>Tada! Here I am!</p>
</div>

</body>
</html>
```

Property Values Forms

CSS1 includes 60 different properties in 7 categories(list can be found in W3C website) Fonts, Lists, Alignment of text, Margins, Colors, Backgrounds, Borders. Keywords property values are used when there are only a few possible values and they are predefined

Eg: small, large, medium. Keyword values are not case sensitive, so Small, SmAlL, and SMALL are all the same as small. Number values can be integer or sequence of digits with decimal points and a + or – sign. Length value are specified as number values that are followed immediately by a two character abbreviation of a unit name. There can be no space between the number and the unit name. The possible unit names are px for pixels, in for inches, cm for centimeters, mm for millimeters, pt for points, pc for picas (12 points),em for value of current font size in pixels, ex for height of the letter _x'. No space is allowed between the number and the unit specification e.g., 1.5 in is illegal!.

Percentage - just a number followed immediately by a percent sign: eg: font size set to 85% means new font size will be 85% of the previous font size value.

URL values: URL property values use a form that is slightly different from references to URLs in links. The actual URL, which can be either absolute or relative, is placed in parentheses and preceded by url, as in the following: url(protocol://server/pathname) No space should be left between URL and the left parenthesis.

Colors : Color name rgb(n1, n2, n3). Hex form: #B0E0E6 stands for powder blue color. Property values are inherited by all nested tags, unless overridden.

Font properties

Font-family The font-family property is used to specify a list of font name. The browser will use the first font in the list that it supports. For example, the following could be specified. fontfamily: Arial, Helvetica, Courier

Generic fonts: They can be specified as the font family value for example :serif, sansserif, cursive, fantasy, and monospace (defined in CSS). Browser has a specific font defined for each generic name. If a font name that has more than one word, it should be single-quoted Eg: font-family: 'Times New Roman' *Font-size* Possible values: a length number or a name, such as smaller, xx-large, medium , large etc. Different browsers can use different relative value for the font-size.
Font-variant The default value of the font-variant property is normal, which specifies the usual character font. This property can be set to small-caps to specify small capital characters.
Font-style The font-style property is most commonly used to specify italic, as in the following example. Eg: font-style: italic
Font-weights The font-weight property is used to specify the degree of boldness. For example: fontweight: bold
Font Shorthands If more than one font property is to be specified than the values may be stated in a list as the value of the font property . The browser will determine from the form of the values which properties to assign. For example, consider the following specification : Eg: font: bold 24pt 'Times New Roman' Palatino Helvetica The order which browser follows is last must be font name, second last font size and then the font style, font variant and font weight can be in any order but before the font size and names. Only the font size and the font family are required in the font value list. Below example displays the fonts.html

```
<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {
   font: 15px arial, sans-serif;
}

p.ex2 {
   font:italic bold 12px/30px Georgia, serif;
}
</style>
</head>
<body>

<p class="ex1">This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.</p>

<p class="ex2">This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.</p>

</body>
</html>
```

Text Decoration
The text-decoration property is used to specify some special features of the text. The available values are line-through, overline, underline, and none, which is the default. Many browsers underline links. Text decoration is not inherited Eg:line-through, overline, underline, none
Eg: p.through {text-decoration: line-through} p.over {text-decoration: overline} p.under {text-decoration: underline}
Letter-spacing – value is any length property value controls amount of space between characters in text

Text Alignment
The text-align property is used to set the horizontal alignment of a text.
A text can be left or right aligned, centered, or justified.
Text Color

The color property is used to set the color of the text.
With CSS, a color is most often specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Text Transformation

- The text-transform property is used to specify uppercase and lowercase letters in a text.
- It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.uppercase {
  text-transform: uppercase;
}
p.lowercase {
  text-transform: lowercase;
}
p.capitalize {
  text-transform: capitalize;
}
</style>
</head>
<body>

<p class="uppercase">This is some text.</p>
<p class="lowercase">This is some text.</p>
<p class="capitalize">This is some text.</p>

</body>
</html>
```

Line Height

The line-height property is used to specify the space between lines:
Text Direction
The direction property is used to change the text direction of an element:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.ex1 {
  direction: rtl;
}
</style>
</head>
<body>

<div>This is default text direction.</div>
<div class="ex1">This is right-to-left text direction.</div>

</body>
```

```
</html>
```

Text Shadow

The text-shadow property adds shadow to text.

The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red):

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
   text-shadow: 3px 2px red;
}
</style>
</head>
<body>

<h1>Text-shadow effect</h1>
<p><b>Note:</b> Internet Explorer 9 and earlier do not support the text-shadow property.</p>

</body>
</html>
```

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

Different List Item Markers

- The list-style-type property specifies the type of list item marker.

```
ul.a {
   list-style-type: circle;
}

ul.b {
   list-style-type: square;
}

ol.c {
   list-style-type: upper-roman;
}

ol.d {
   list-style-type: lower-alpha;
}
```

- The list-style-image property specifies an image as the list item marker

```
ul {
    list-style-image: url('sqpurple.gif');
}
```

- The list-style-position property specifies whether the list-item markers should appear inside or outside the content flow

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

- The list-style property is a shorthand property. It is used to set all the list properties in one declaration.

```
ul {
    list-style: square inside url("sqpurple.gif");
}
```

<u>Colors in CSS are most often specified by:</u>

- a valid color name - like "red"
- an RGB value - like "rgb(255, 0, 0)"
- a HEX value - like "#ff0000"

The CSS background properties are used to define the background effects for elements.

<u>CSS background properties:</u>

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

<u>Background properties</u>

- The background-color property specifies the background color of an element.
- body {
    background-color: lightblue;
  }
- The background-image property specifies an image to use as the background of an element.
- body {
    background-image: url("paper.gif");
  }
- The background-repeat property specifies image to repeat horizontal(repeat-x) or vertical(repeat-y) or no repeat

- The position of the image is specified by the background-position property
- To specify that the background image should be fixed (will not scroll with the rest of the page), use the background-attachment property:

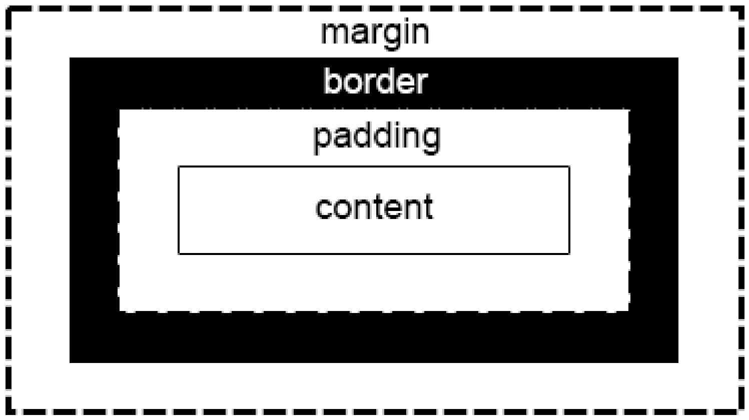
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
    background-position: right top;
    margin-right: 200px;
    background-attachment: fixed;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
</body>
</html>
```

CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

```
div {
    width: 300px;
    border: 25px solid green;
    padding: 25px;
    margin: 25px;
}
```

CSS Layout - The position Property

The position property specifies the type of positioning method used for an element (static, relative, fixed or absolute).

The position Property

The position property specifies the type of positioning method used for an element.

There are four different position values:

- static
- relative
- fixed
- absolute

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This <div> element has position: static;

Here is the CSS that is used:

Example

```
div.static {
  position: static;
  border: 3px solid #73AD21;
}
```

position: relative;

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

This <div> element has position: relative;

Here is the CSS that is used:

Example

```
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```

position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example

```
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
```

```
    width: 300px;
    border: 3px solid #73AD21;
}
```

This <div> element has position: fixed;

position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except static.

Here is a simple example:

This <div> element has position: relative;

This <div> element has position: absolute;

Here is the CSS that is used:

Example

```
div.relative {
    position: relative;
    width: 400px;
    height: 200px;
    border: 3px solid #73AD21;
}

div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21;
}
```

Overlapping Elements

When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative stack order:

This is a heading

Because the image has a z-index of -1, it will be placed behind the text.

Example

```
img {
    position: absolute;
    left: 0px;
    top: 0px;
    z-index: -1;
}
```

An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

Positioning Text In an Image

How to position text over an image:

Example

Bottom Left

Top Left

Top Right

Bottom Right

CSS Layout - float and clear

The float property specifies whether or not an element should float.

The clear property is used to control the behavior of floating elements.

The float Property

In its simplest use, the float property can be used to wrap text around images.

The following example specifies that an image should float to the right in a text:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
    float: right;
    margin: 0 0 10px 10px;
}
</style>
</head>
<body>
<p>In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.</p>
<p><img src="w3css.gif" alt="W3Schools.com" width="100" height="140">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae
```

massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, ………...Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p>
</body>
</html>

The clear Property

The clear property is used to control the behavior of floating elements.
Elements after a floating element will flow around it. To avoid this, use the clear property.
The clear property specifies on which sides of an element floating elements are not allowed to float:

```
<!DOCTYPE html>
<html>
<head>
<style>
.div1 {
  float: left;
  width: 100px;
  height: 50px;
  margin: 10px;
  border: 3px solid #73AD21;
}

.div2 {
  border: 1px solid red;
}


.div3 {
  float: left;
  width: 100px;
  height: 50px;
  margin: 10px;
  border: 3px solid #73AD21;
}

.div4 {
  border: 1px solid red;
  clear: left;
}
</style>
</head>
<body>

<h2>Without clear</h2>
<div class="div1">div1</div>
<div class="div2">div2 - Notice that the div2 element is after div1, in the HTML code. However, since div1 is floated to the left, this happens: the text in div2 is floated around div1, and div2 surrounds the whole thing.</div>

<h2>Using clear</h2>
<div class="div3">div3</div>
<div class="div4">div4 - Using clear moves div4 down below the floated div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".</div>
```

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

Conflict Resolution

When two or more rules apply to the same tag there are resolutions for deciding which rule applies. In-line style sheets have precedence over document style sheets. Document style sheets have precedence over external style sheets. Within the same level there can be conflicts a tag may be used twice as a selector
h3{color:red;} body h3 {color: green;} A tag may inherit a property and also be used as a selector. Style sheets can have different sources:
The browser itself may set some style eg: In FX2 min font size can be set in Tools-Options-Advanced window The author of a document may specify styles. The user, through browser settings, may specify styles. Individual properties can be specified as important or normal.
Eg: p.special{font-style: italic !important; font-size :14}
This property is known as weight of a specification. Conflict resolution is a multistage sorting process. The first step in the process is to gather the style specifications from the three possible levels of style sheets. These specifications are sorted into order by the relative precedence of the style sheet levels. This is done according to the following rules, in which the first has the highest precedence. From highest to lowest
1. Important declarations with user origin
2. Important declarations with author origin
3. Normal declarations with author origin
4. Normal declarations with user origin
5. Any declarations with browser (or other user agent) origin Tie-Breakers
Conflict resolution by Specificity (high to low)
1. id selectors
2. Class and pseudo-class selectors
3. Contextual selectors
4. General selectors
Position Essentially, later has precedence over earlier. Most recently seen specification is the one which gets more precedence. Sorting process to resolve the style specification is known as cascade.

Comparison between SGML and HTML:

	SGML	HTML
Full Form	It stands for the Standard Generalized Markup Language.	It stands for Hyper Text Markup Language.
Type	application/sgml, text/sgml	text/html
Type code	Text	Text
Uniform type	public.xml	public.html
Developed by	ISO	WWW Consotium
Format type	It is a mark up language.	It is a mark up language.
Extended from	GML	SGML
Extended to	HTML,XML	XHTML

Comparison between html and xhtml

	HTML	XHTML
Introduction (from	HTML or HyperText Markup Language is the main markup	XHTML (Extensible HyperText Markup Language) is a family of XML markup languages that mirror or extend

Wikipedia)	language for creating web pages and other information that can be displayed in a web browser.	versions of the widely used Hypertext Markup Language (HTML), the language in which web pages are written.
Filename extension	.html, .htm	.xhtml, .xht, .xml, .html, .htm
Internet media type	text/html	application/xhtml+xml
Developed by	W3C & WHATWG	World Wide Web Consortium
Type of format	Document file format	Markup language
Extended from	SGML	XML, HTML
Stands for	HyperText Markup Language	Extensible HyperText Markup Language
Application	Application of Standard Generalized Markup Language (SGML).	Application of XML
Function	Web pages are written in HTML.	Extended version of HTML that is stricter and XML-based.
Nature	Flexible framework requiring lenient HTML specific parser.	Restrictive subset of XML and needs to be parsed with standard XML parsers.
Origin	Proposed by Tim Berners-Lee in 1987.	World Wide Web Consortium Recommendation in 2000.
Versions	HTML 2, HTML 3.2, HTML 4.0, HTML 5.	XHTML 1, XHTML 1.1, XHTML 2, XHTML 5.

Chapter-4
JavaScript

Overview of Javascript

JavaScript is a sequence of statements to be executed by the browser. It is most popular scripting language on the internet, and works in all major browsers, such as IE, FireFox, chrome, opera safari. Prequisite – HTML/XHTML

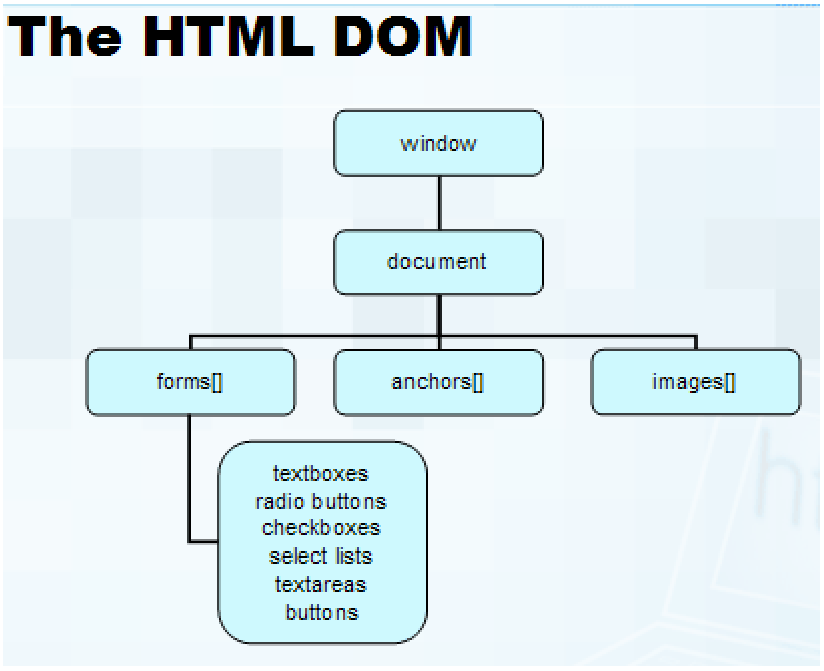
Origins

It is originally known as LiveScript, developed by Netscape. It become a joint venture of Netscape and Sun in 1995, and was renamed as JavaScript. It was standardized by the European computer Manufacturers Association as ECMA-262. ISO-16262. Current standard specifications can be found at http://www.ecma-international.org/publications/standardsEcma-262.htm Collections of JavaScript code scripts and not programs.

What is JavaScript?

1. JavaScript was designed to add interactivity to HTML pages.
2. JavaScript is a scripting language.
3. A scripting language is a lightweight programming language.
4. It is usually embedded directly into HTML pages.
5. JavaScript is an interpreted language (Scripts are executed without preliminary compilations) **JavaScript can be divided into three parts.**

1. The Core: It is a heart of the language, including its operators, expressions, statements and subprograms.
2. Client Side: It is a collection of objects that support control of a browser and interactions with users. Eg. With JavaScript an XHTML document can be made to be responsible to user inputs. Such as mouse clicks and keyboard use.
3. Server side: It is a collection of objects that make the language useful on a Web server. Eg. To support communication with a DBMS. Client side JavaScript is an XHTML embedded scripting language. We refer to every collection of JavaScript code as a script. An XHTML document can include any number of embedded scripts. The HTML Document Object Model(DOM) is the browsers view of an HTML page as an object hierarchy, starting with the browser window itself and moving deeper into the page, including of the elements on the page and their attribute.



The top level object is window. The document object is a child of window and all the objects that appear on the page are descendants of the document object. These objects can have children of their own. Eg. Form objects generally have several child objects , including textboxes, radio buttons and select menus.
DOM allows JS to access and modify the CSS properties and content of any element of a displayed XHTML document

Java vs JavaScript

JavaScript and java is only related through syntax.
JavaScript support for OOP is different from that of Java.
JavaScript is dynamically typed.
Java is strongly typed language. Types are all known at compile time and operand types are checked for compatibility. But variables in JavaScript need not be declared and are dynamically typed, making compile time type checking impossible.
Objects in Java are static -> their collection of data number and methods is fixed at compile time. JavaScript objects are dynamic: The number of data members and methods of an object can change during execution.

Java	JavaScript
Java is an Object Oriented Programming (OOP) language created by James Gosling of Sun Microsystems (later acquired by Oracle Corporation).	JavaScript is a client-side (OOP) scripting language that was created by Netscape and was originally known as LiveScript.
Java programs require Java Virtual Machine (JVM) for execution.	JavaScript requires to be embedded within HTML which is then executed by a web browser, such as Mozilla Firefox.
Java programs need to be compiled into "byte code" before they can be executed by the JVM. Therefore, a compilation is needed every time a change is made to the code.	JavaScript is plain text-based programming which is directly executed by a scripting engine within a browser.Therefore, change in the code does not need any conversion or compilation and can be executed directly.
Enabling Java in a browser does not mean that you have enabled JavaScript too.	Enabling JavaScript in a browser does not mean that you have enabled Java too.
Java implements static typing of data types.	Java implements dynamic typing of data types.
Java is a strongly-typed language.	Java is a weakly-typed language.
Objects in the Java language are class-based.	Objects in the JavaScript language are prototype-based.
Java source code is well hidden because it is compiled into an unreadable byte code.	JavaScript code can be read by any person because it is in a plain text form.
To make Java programs, you will probably require an entire environment, that's Java Development Kit (JDK).	JavaScript can simply be written using any text editor.
Java is engineered keeping "security" as the prime priority. This is achieved using the JVM.	JavaScript is less secure compared to Java. However, there are efforts to make it secure from things like cross-site-scripting (XSS), etc.

Uses of JavaScript:

Goal of JavaScript is to provide programming capability at both server and the client ends of a Web connection. Client-side JavaScript is embedded in XHTML documents and is interpreted by the browser. This transfer of load from the often overloaded server to the normally under loaded client can obviously benefit all other clients. It cannot replace server side computations like file operations, database access, and networking. JavaScript can be used as an alternative to Java applets. Java applets are downloaded separately from the XHTML documents that call them but JavaScript are integral part of XHTML document, so no secondary downloading is necessary. Java applets far better for graphics files scripts. Interactions with users through form elements, such as buttons and menus, can be conveniently described in JavaScript. Because events such as button clicks and mouse movements are easily detected with JavaScript they can be used to trigger computations and provide feedback to the users. Eg. When user moves the mouse cursor from a textbox, JavaScript can detect that movement and check the appropriateness of the text box's value. Even without forms, user interactions are both possible and simple to program. These interactions which take place in dialog windows include getting input from the user and allowing the user to make choices through buttons. It is also easy to generate new content in the browser display dynamically.

Event driven computation Event driven computation means that the actions often are executed in response to actions often are executed in response to actions of the users of doc, actions like mouse clicks and form submissions. This type of computation supports user interactions through XHTML form elements on the client display. One of the common uses of JS is client end input data validation values entered by users will be checked before sending them to server for further processing. This becomes more efficient to perform input data checks and carry on this user dialog entirely on the client. This saves both server time and internet time.

Browsers and XHTML/JS documents. It is an XHTML document does not include embedded scripts, the browser reads the lines of the document and renders its window according to the tags, attributes and content it finds when a JavaScript script is encountered in the doc, the browser uses its JS interpreter to execute the script. When the end of script reached, the browser goes back to reading the XHTML document and displaying its content. JS scripts can appear in either part of an XHTML document, the head or the body, depending on the purpose of the script. Scripts that produce content only when requested or that react to user interactions are placed in the head of the document. -> Function definition and code associated with form elements such as buttons. Scripts that are to be interpreted just once, when the interpreter finds them are placed in the document body. Accordingly, the interpreter notes the existence of scripts that appear in the head of a document, but it does not interpret them while processing the head. Scripts that are found in the body of a document are interpreted as they are found.

Object orientation and Javascript JavaScript is object based language. It doesn't have classes. Its objects serve both as objects and as models of objects. JavaScript does not support class based inheritance as is supported in OO language. CTT-Java. But it supports prototype based inheritance i.e a technique that can be used to simulate some of the aspects of inheritance. JavaScript does not support polymorphism. A polymorphic variable can reference related objects of different classes within the same class hierarchy. A method call through such a polymorphic variable can be dynamically bound to the method in the objects class.
JavaScript Objects JavaScript objects are collection of prospectus, which corresponds to the members of classes in Java & C++. Each property is either a data property or a function or method property.
1. Data Properties a. Primitive Values (Non object Types) b. Reference to other objects
2. Method Properties –methods.

<u>Primitives </u>are non object types and are used as they can be implemented directly in hardware resulting in faster operations on their values. These are accessed directly-like scalar types in java & C++ called value types. All objects in a JavaScript programs are directly accessed through variables. Such a variable is like a reference in java. The properties of an object are referenced by attaching the name of the property to the variable that references the object.
Eg. If myCar variable referencing an object that has the property engine, the engine property can be referenced with myCar.engine.

The root object in JavaScript is object. It is ancestor through prototype inheritance, of all objects. Object is most generic of all objects, having some methods but no data properties. All other objects are specializations of object, and all inherit its methods. JavaScript object appears both internally and externally as a list of property/value pairs. Properties are names values are data values of functions. All functions are objects and are referenced through variables. The collection of properties of JavaScript is dynamic – Properties can be added or deleted at any time.

General syntactic Characteristics
1. JavaScript are embedded either directly or indirectly in XHTML documents.
2. Scripts can appear directly as the content of a <style> tag
3. the type attribute of <style> tag must be set to text/JavaScript.
4. the JavaScript can be indirectly embedded in an XHTML document using the src attribute of a <script>tag, whose value is name of a file that contains the scripts.
Closing tag is required even if script element has src attribute included.
The indirect method of embedding JavaScript in XHTML has advantages of
1) Hiding the script from the browser user.
2) It also avoids the problem of hiding scripts from older browsers.
3) It is good to separate the computation provided by JavaScript from the layout and presentation provided by XHTML and CSS respectively. But it is sometimes not convenient and cumbersome to place all JavaScript code in separate file JavaScript identifiers or names are similar to programming languages. 1. must begin with (-), or a letter. Subsequent characters may be letters, underscores or digits.
2. No length limitations for identifiers.
3. Case sensitive 4. No uppercase letters.
Reserved words are break delete function return typeof case do if switch var catch else in this void continue finally instanceof throw while default for new try with *JavaScript has large collection of predefined words* alert, open, java, self
Comments in JavaScript
// - Single line /* */ -Multiple line
Two issues regarding embedding JavaScript in XHTML documents.
1) There are some browsers still in use that recognize the <script > tag but do not have JS interpreters. These browsers will ignore the contents of the script element and cause no problems.
2) There are still a few browsers in use that are so old they do not recognize <script> tag These browsers will display the contents of the script elements as if it were just text. Therefore it has been customary to enclose the contents of all script elements in XHTML comments to avoid this problem. XHTML validator also has a problem with embedded JS. When embedded JS happens to include recognizable tags.
For example <br/> in output of JS-they often cause validation errors. Therefore we have to enclose embedded JS in XHTML comments. XHTML comment introduction (<!--) works as a hiding prelude to JS code. Syntax for closing a comment that encloses JS code is different. It is usual XHTML comment closer but it must be on its own line and preceeded by two slashes.
Eg: <!—
---js –
//-->
Many more problem are associated with putting embedded JavaScript in comments in XHTML document. Solution : Put JavaScript scripts of significant style in separate files. Use of ; in JS is unusual When EOL coincides with end of statement, the interpreter effectively insects a semicolon there, but this leads to problems.
Eg. return x;
Interpreter puts; after return making x an illegal orphan. Therefore put JS statements on its own line when possible and terminate each statement with a semicolon. If stmt does not fit in one line, break the stmt at a place that will ensure that the first line does not have the form of a complete statement.

```
<html>
  <body>
```

```
      <script language="javascript" type="text/javascript">
        <!--
          document.write("Hello World!")
        //-->
      </script>
   </body>
</html>
```

Primitives, operations, and expressions

The primitive data types, operations and expressions of JavaScript.

Primitive Types:

Pure primitive types : Number, String, Boolean, Undefined and null. JavaScript includes predefined objects that are closely related to the number, string and Boolean types named number, string and Boolean. These are wrapper objects. Each contains a property that stores a value of the corresponding primitive type. The purpose of the wrapper object is to provide properties and methods that are convenient for use with values of the primitive types.

In case of numbers : Properties are more useful.

In case of string : Methods are more useful.

Because JavaScript coerces values between the number type and number objects and between the string type and objects, the methods of number and string can be used on variables of the corresponding primitive types.

Numeric and String literals: All numeric literals are values of type number. The numeric values of JavaScript are represented internally in double precision floating point form, Numeric values in JavaScript are called numbers because of single numeric data type. Literal numbers in a script can have forms of either integers or floating point values. Integer literals are strings of digits. Floating point literals can have decimal points or exponents or both.

Legal numeric literals: 72, 7.2, .72, 72, 7E2, 7e2, .7e2, 7.e2, 7.2E-2.

Integers in Hexadecimal form 0x or 0X.String Literal: Sequence of 0 or more characters delimited by either single quotes or double quotes. They can include characters specified with escape sequences, such as \n and \t. If you want an actual single quote character in a string literal that is delimited by single quotes, embedded single quote must be preceded by a backslash

Screen output and keyboard input

A JavaScript is interpreted when the browser finds the script in the body of the XHTML document. Thus the normal screen for the JavaScript is the same as the screen in which the content of the host XHTML document is displayed. JS models the XHTML document with the document object. The window in which the browser displays an XHTML document is modeled with the window object. It includes two properties document and window.

Document -> Refers to document object.

Window -> self referential and refers to window object.

Methods -> write

Write is used to create script o/p, which is dynamically created XHTML document content.

Eg: document.write("the result:",result,"<br/>");

Window object is JS model for the browser window. It includes three methods that create dialog boxes for three specific kinds of user interactions. The default object for JS is window object currently being displayed, so calls to these methods need not include an object reference. Alert method opens a dialog window and displays its parameter in that window. It also displays an OK button. The parameter string to alert is not XHTML code, it is plain text. Therefore the string parameter to alert may include \n but never should include <br/>

Confirm method opens a dialog window in which it displays its string parameter, along with two buttons OK and Cancel. Confirm returns a Boolean value that indicates the users button input

True -> for OK

False-> for cancel.

Eg: var question=confirm("do you want continue this download"?);

After the user responds to one of the button in the confirm dialog window script can test the variable, question and react accordingly. Prompt method creates a dialog window that contains a text box which is used to collect a string of input from the user, which prompt returns as its value. The window also includes two buttons, OK and Cancel, prompt takes two parameters the string that prompts the user for input and a default string in case the user does not type a string before pressing one of the two buttons. In many cases an empty string is used for the default input.
Eg. name name=prompt("What is your name","");
Alert, prompt and confirm cause the browser to wait for a user response.
Alert – OK
Prompt-OK, Cancel
Confirm-OK, Cancel.

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

<p>Click the button to demonstrate the prompt box.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
    var person = prompt("Please enter your name", "Harry Potter");
    if (person != null) {
        document.getElementById("demo").innerHTML =
        "Hello " + person + "! How are you today?";
    }
}
</script>

</body>
</html>
```

Element access in Javascript
Javascript provides the ability for getting the value of an element on a webpage as well as dynamically changing the content within an element.

Getting the value of an element
To get the value of an element, the getElementById method of the document object is used. For this method to get the value of an element, that element has to have an id given to it through the id attribute

Control Statements
Control expression control the order of execution of statements. Compound statements is JavaScript are syntactic contains for sequences of statements whose execution they control. Compound statement sequence of statements deleted by braces. Control construct is a control statement whose execution it controls. Compound statements are not allowed to create local variables.
Control Expressions. Control statements flow.
Eg. primitive values, relational expression and compound expressions. Result of evaluating a control expression is Boolean value true or false.
For strings. true - string false-null string

For number True- any number

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	**Description**
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	**Example**	**Same As**
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

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

JavaScript supports the following forms of if..else statement –

if statement

if...else statement

if...else if... statement.

if statement

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax

The syntax for a basic if statement is as follows –

```
if (expression){
  Statement(s) to be executed if expression is true
}
```

if...else statement:

The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax

```
if (expression){
  Statement(s) to be executed if expression is true
}

else{
  Statement(s) to be executed if expression is false
}
```

if...else if... statement

The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.

Syntax

The syntax of an if-else-if statement is as follows –

```
if (expression 1){
  Statement(s) to be executed if expression 1 is true
}

else if (expression 2){
  Statement(s) to be executed if expression 2 is true
}

else if (expression 3){
  Statement(s) to be executed if expression 3 is true
}

else{
  Statement(s) to be executed if no expression is true
}
```

The objective of a switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

```
switch (expression)
{
  case condition 1: statement(s)
  break;

  case condition 2: statement(s)
  break;
  ...

  case condition n: statement(s)
  break;

  default: statement(s)
}
```

The break statements indicate the end of a particular case. If they were omitted, the interpreter would continue executing each statement in each of the following cases.

```
<html>
  <body>

    <script type="text/javascript">
      <!--
        var grade='A';
        document.write("Entering switch block<br />");
        switch (grade)
        {
          case 'A': document.write("Good job<br />");
          break;
```

```
            case 'B': document.write("Pretty good<br />");
            break;

            case 'C': document.write("Passed<br />");
            break;

            case 'D': document.write("Not so good<br />");
            break;

            case 'F': document.write("Failed<br />");
            break;

            default:  document.write("Unknown grade<br />")
         }
         document.write("Exiting switch block");
      //-->
    </script>

    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

The syntax of while loop in JavaScript is as follows −

```
while (expression){
  Statement(s) to be executed if expression is true
}
```

The syntax for do-while loop in JavaScript is as follows −

```
do{
  Statement(s) to be executed;
} while (expression);
```

The syntax of for loop is JavaScript is as follows −

```
for (initialization; test condition; iteration statement){
  Statement(s) to be executed if test condition is true
}
```

The for...in loop is used to loop through an object's properties. As we have not discussed Objects yet, you may not feel comfortable with this loop. But once you understand how objects behave in JavaScript, you will find this loop very useful.

Syntax

```
for (variablename in object){
  statement or block to execute
}
```

In each iteration, one property from object is assigned to variablename and this loop continues till all the properties of the object are exhausted.

```
<html>
  <body>

    <script type="text/javascript">
      <!--
        var aProperty;
```

```
            document.write("Navigator Object Properties<br /> ");

            for (aProperty in navigator) {
              document.write(aProperty);
              document.write("<br />");
            }
            document.write ("Exiting from the loop!");
          //-->
        </script>

        <p>Set the variable to different object and then try...</p>
    </body>
</html>
```

JavaScript provides full control to handle loops and switch statements. There may be a situation when you need to come out of a loop without reaching its bottom. There may also be a situation when you want to skip a part of your code block and start the next iteration of the loop.

To handle all such situations, JavaScript provides break and continue statements. These statements are used to immediately come out of any loop or to start the next iteration of any loop respectively.

The break Statement

The break statement, which was briefly introduced with the switch statement, is used to exit a loop early, breaking out of the enclosing curly braces.

The continue Statement

The continue statement tells the interpreter to immediately start the next iteration of the loop and skip the remaining code block. When a continue statement is encountered, the program flow moves to the loop check expression immediately and if the condition remains true, then it starts the next iteration, otherwise the control comes out of the loop.

```
<html>
    <body>
         <script type="text/javascript">
          <!--
            var x = 1;
            document.write("Entering the loop<br /> ");

            while (x < 10)
            {
              x = x + 1;

              if (x == 5){
                continue; // skip rest of the loop body
              }
              document.write( x + "<br />");
            }
          document.write("Exiting the loop!<br /> ");
          //-->
        </script>

        <p>Set the variable to different value and then try...</p>
    </body>
</html>
```

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions. You must have seen functions like alert() and write() in the earlier chapters. We were using these functions again and again, but they had been written in core JavaScript only once.

JavaScript allows us to write our own functions as well.
Function Definition
Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax
The basic syntax is shown here.

```
<script type="text/javascript">
  <!--
    function functionname(parameter-list)
    {
      statements
    }
  //-->
</script>
```
Example
Try the following example. It defines a function called sayHello that takes no parameters −

```
<script type="text/javascript">
  <!--
    function sayHello()
    {
      alert("Hello there");
    }
  //-->
</script>
```
Calling a Function
To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
  <head>

    <script type="text/javascript">
      function sayHello()
      {
        document.write ("Hello there!");
      }
    </script>

  </head>
  <body>
```

```
      <p>Click the following button to call the function</p>

      <form>
         <input type="button" onclick="sayHello()" value="Say Hello">
      </form>

      <p>Use different text in write method and then try...</p>
   </body>
</html>
```

Output

Function Parameters

Till now, we have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

Example

Try the following example. We have modified our sayHello function here. Now it takes two parameters.

```
<html>
   <head>

      <script type="text/javascript">
         function sayHello(name, age)
         {
            document.write (name + " is " + age + " years old.");
         }
      </script>

   </head>
   <body>
      <p>Click the following button to call the function</p>

      <form>
         <input type="button" onclick="sayHello('Zara', 7)" value="Say Hello">
      </form>
         <p>Use different parameters inside the function and then try...</p>
   </body>
</html>
```

The return Statement

A JavaScript function can have an optional return statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.

For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

Example

Try the following example. It defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

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

```
<script type="text/javascript">
  function concatenate(first, last)
  {
    var full;
    full = first + last;
    return full;
  }

  function secondFunction()
  {
    var result;
    result = concatenate('Zara', 'Ali');
    document.write (result );
  }
</script>
 </head>
<body>
<p>Click the following button to call the function</p>
  <form>
   <input type="button" onclick="secondFunction()" value="Call Function">
</form>
  <p>Use different parameters inside the function and then try...</p>
 </body>
</html>
```

JavaScript is an Object Oriented Programming (OOP) language. A programming language can be called object-oriented if it provides four basic capabilities to developers −

- **Encapsulation** − the capability to store related information, whether data or methods, together in an object.
- **Aggregation** − the capability to store one object inside another object.
- **Inheritance** − the capability of a class to rely upon another class (or number of classes) for some of its properties and methods.
- **Polymorphism** − the capability to write one function or method that works in a variety of different ways.

Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object, otherwise the attribute is considered a property.

Object Properties

Object properties can be any of the three primitive data types, or any of the abstract data types, such as another object. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that are used throughout the page.

The syntax for adding a property to an object is −

```
objectName.objectProperty = propertyValue;
```

For example − The following code gets the document title using the "title" property of the document object.

```
var str = document.title;
```

Object Methods

Methods are the functions that let the object do something or let something be done to it. There is a small difference between a function and a method – at a function is a standalone unit of statements and a method is attached to an object and can be referenced by the this keyword.

Methods are useful for everything from displaying the contents of the object to the screen to performing complex mathematical operations on a group of local properties and parameters.

For example − Following is a simple example to show how to use the write() method of document object to write any content on the document.

```
document.write("This is test");
```

User-Defined Objects

All user-defined objects and built-in objects are descendants of an object called Object.

The new Operator

The new operator is used to create an instance of an object. To create an object, the new operator is followed by the constructor method.

In the following example, the constructor methods are Object(), Array(), and Date(). These constructors are built-in JavaScript functions.

```
var employee = new Object();
var books = new Array("C++", "Perl", "Java");
var day = new Date("August 15, 1947");
```

The Object() Constructor

A constructor is a function that creates and initializes an object. JavaScript provides a special constructor function called Object() to build the object. The return value of the Object() constructor is assigned to a variable.

The variable contains a reference to the new object. The properties assigned to the object are not variables and are not defined with the var keyword.

Example 1

Try the following example; it demonstrates how to create an Object.

```
<html>
  <head>
    <title>User-defined objects</title>

    <script type="text/javascript">
      var book = new Object();   // Create the object
      book.subject = "Perl"; // Assign properties to the object
      book.author  = "Mohtashim";
    </script>
  </head>

  <body>
      <script type="text/javascript">
      document.write("Book name is : " + book.subject + "<br>");
      document.write("Book author is : " + book.author + "<br>");
    </script>
    </body>
</html>
```

Example 2

This example demonstrates how to create an object with a User-Defined Function. Here this keyword is used to refer to the object that has been passed to a function.

```
<html>
  <head>

  <title>User-defined objects</title>
      <script type="text/javascript">
      function book(title, author){
        this.title = title;
        this.author  = author;
      }
    </script>
 </head>
  <body>
      <script type="text/javascript">
      var myBook = new book("Perl", "Mohtashim");
      document.write("Book title is : " + myBook.title + "<br>");
      document.write("Book author is : " + myBook.author + "<br>");
    </script>
  </body>
</html>
```

Defining Methods for an Object
The previous examples demonstrate how the constructor creates the object and assigns properties. But we need to complete the definition of an object by assigning methods to it.

Example
Try the following example; it shows how to add a function along with an object.

```
<html>
  <head>
  <title>User-defined objects</title>

    <script type="text/javascript">
      // Define a function which will work as a method
      function addPrice(amount){
        this.price = amount;
      }

      function book(title, author){
        this.title = title;
        this.author  = author;
        this.addPrice = addPrice; // Assign that method as property.
      }
    </script>
  </head>
  <body>

    <script type="text/javascript">
      var myBook = new book("Perl", "Mohtashim");
      myBook.addPrice(100);

      document.write("Book title is : " + myBook.title + "<br>");
      document.write("Book author is : " + myBook.author + "<br>");
      document.write("Book price is : " + myBook.price + "<br>");
    </script>

  </body>
</html>
```

The 'with' Keyword

The 'with' keyword is used as a kind of shorthand for referencing an object's properties or methods.
The object specified as an argument to with becomes the default object for the duration of the block that follows. The properties and methods for the object can be used without naming the object.

Syntax

The syntax for with object is as follows −

```
with (object){
  properties used without the object name and dot
}
```

Example

Try the following example.

```
<html>
  <head>
  <title>User-defined objects</title>

    <script type="text/javascript">
      // Define a function which will work as a method
      function addPrice(amount){
        with(this){
          price = amount;
        }
      }

      function book(title, author){
        this.title = title;
        this.author  = author;
        this.price = 0;
        this.addPrice = addPrice; // Assign that method as property.
      }
    </script>

  </head>
  <body>

    <script type="text/javascript">
      var myBook = new book("Perl", "Mohtashim");
      myBook.addPrice(100);

      document.write("Book title is : " + myBook.title + "<br>");
      document.write("Book author is : " + myBook.author + "<br>");
      document.write("Book price is : " + myBook.price + "<br>");
    </script>
  </body>
</html>
```

Arrays

JavaScript arrays are used to store multiple values in a single variable.
It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Syntax

Use the following syntax to create an Array object −

```
var fruits = new Array( "apple", "orange", "mango" );
```

The Array parameter is a list of strings or integers. When you specify a single numeric parameter with the Array constructor, you specify the initial length of the array. The maximum length allowed for an array is 4,294,967,295.

You can create array by simply assigning values as follows −

```
var fruits = [ "apple", "orange", "mango" ];
```

You will use ordinal numbers to access and to set values inside an array as follows.

fruits[0] is the first element
fruits[1] is the second element
fruits[2] is the third element

Array Properties

Here is a list of the properties of the Array object along with their description.

Property	Description
constructor	Returns a reference to the array function that created the object.
index	The property represents the zero-based index of the match in the string
input	This property is only present in arrays created by regular expression matches.
length	Reflects the number of elements in an array.
prototype	The prototype property allows you to add properties and methods to an object.

Array Methods

Here is a list of the methods of the Array object along with their description.

Method	Description
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
every()	Returns true if every element in this array satisfies the provided testing function.
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.

join()	Joins all elements of an array into a string.
lastIndexOf()	Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
map()	Creates a new array with the results of calling a provided function on every element in this array.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reduce()	Apply a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.
reduceRight()	Apply a function simultaneously against two values of the array (from right-to-left) as to reduce it to a single value.
reverse()	Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that element.
slice()	Extracts a section of an array and returns a new array.
some()	Returns true if at least one element in this array satisfies the provided testing function.
toSource()	Represents the source code of an object
sort()	Sorts the elements of an array
splice()	Adds and/or removes elements from an array.
toString()	Returns a string representing the array and its elements.
unshift()	Adds one or more elements to the front of an array and returns the new length of the array.

Events:

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc. Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

onclick Event Type

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

Example

Try the following example.

```
<html>
  <head>

    <script type="text/javascript">
      <!--
```

```
            function sayHello() {
               alert("Hello World")
            }
         //-->
      </script>

   </head>

   <body>
      <p>Click the following button and see result</p>

      <form>
         <input type="button" onclick="sayHello()" value="Say Hello" />
      </form>

   </body>
</html>
```

onsubmit Event type

onsubmit is an event that occurs when you try to submit a form. You can put your form validation against this event type.

Example

The following example shows how to use onsubmit. Here we are calling a validate() function before submitting a form data to the webserver. If validate() function returns true, the form will be submitted, otherwise it will not submit the data.

Try the following example.

```
<html>
   <head>

      <script type="text/javascript">
         <!--
            function validation() {
               all validation goes here
               .........
               return either true or false
            }
         //-->
      </script>

   </head>
   <body>

      <form method="POST" action="t.cgi" onsubmit="return validate()">
         .......
         <input type="submit" value="Submit" />
      </form>

   </body>
</html>
```

onmouseover and onmouseout

These two event types will help you create nice effects with images or even with text as well. The onmouseover event triggers when you bring your mouse over any element and the onmouseout triggers when you move your mouse out from that element. Try the following example.

```
<html>
  <head>

    <script type="text/javascript">
      <!--
        function over() {
          document.write ("Mouse Over");
        }

        function out() {
          document.write ("Mouse Out");
        }

      //-->
    </script>

  </head>
  <body>
    <p>Bring your mouse inside the division to see the result:</p>

    <div onmouseover="over()" onmouseout="out()">
      <h2> This is inside the division </h2>
    </div>

  </body>
</html>
```

Common HTML Events

Here is a list of some common HTML events:

Event	**Description**
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

The HTML DOM allows JavaScript to change the style of HTML elements.

Changing HTML Style
To change the style of an HTML element, use this syntax:

```
document.getElementById(id).style.property = new style
```

Using Events
The HTML DOM allows you to execute code when an event occurs.

Events are generated by the browser when "things happen" to HTML elements:

An element is clicked on
The page has loaded
Input fields are changed

This example changes the style of the HTML element with id="id1", when the user clicks a button:

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

<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>

</body>
</html>
```

HTML 5 Standard Events

The standard HTML 5 events are listed here for your reference. Here script indicates a Javascript function to be executed against that event.

Attribute	Value	Description
Offline	script	Triggers when the document goes offline
Onabort	script	Triggers on an abort event
onafterprint	script	Triggers after the document is printed
onbeforeonload	script	Triggers before the document loads
onbeforeprint	script	Triggers before the document is printed
onblur	script	Triggers when the window loses focus
oncanplay	script	Triggers when media can start play, but might has to stop for buffering
oncanplaythrough	script	Triggers when media can be played to the end, without stopping for buffering
onchange	script	Triggers when an element changes
onclick	script	Triggers on a mouse click
oncontextmenu	script	Triggers when a context menu is triggered
ondblclick	script	Triggers on a mouse double-click
ondrag	script	Triggers when an element is dragged
ondragend	script	Triggers at the end of a drag operation
ondragenter	script	Triggers when an element has been dragged to a valid drop target
ondragleave	script	Triggers when an element is being dragged over a valid drop target
ondragover	script	Triggers at the start of a drag operation
ondragstart	script	Triggers at the start of a drag operation

ondrop	script	Triggers when dragged element is being dropped
ondurationchange	script	Triggers when the length of the media is changed
onemptied	script	Triggers when a media resource element suddenly becomes empty.
onended	script	Triggers when media has reach the end
onerror	script	Triggers when an error occur
onfocus	script	Triggers when the window gets focus
onformchange	script	Triggers when a form changes
onforminput	script	Triggers when a form gets user input
onhaschange	script	Triggers when the document has change
oninput	script	Triggers when an element gets user input
oninvalid	script	Triggers when an element is invalid
onkeydown	script	Triggers when a key is pressed
onkeypress	script	Triggers when a key is pressed and released
onkeyup	script	Triggers when a key is released
onload	script	Triggers when the document loads
onloadeddata	script	Triggers when media data is loaded
onloadedmetadata	script	Triggers when the duration and other media data of a media element is loaded
onloadstart	script	Triggers when the browser starts to load the media data
onmessage	script	Triggers when the message is triggered
onmousedown	script	Triggers when a mouse button is pressed
onmousemove	script	Triggers when the mouse pointer moves
onmouseout	script	Triggers when the mouse pointer moves out of an element
onmouseover	script	Triggers when the mouse pointer moves over an element
onmouseup	script	Triggers when a mouse button is released
onmousewheel	script	Triggers when the mouse wheel is being rotated
onoffline	script	Triggers when the document goes offline
onoine	script	Triggers when the document comes online
ononline	script	Triggers when the document comes online
onpagehide	script	Triggers when the window is hidden
onpageshow	script	Triggers when the window becomes visible
onpause	script	Triggers when media data is paused
onplay	script	Triggers when media data is going to start playing
onplaying	script	Triggers when media data has start playing
onpopstate	script	Triggers when the window's history changes
onprogress	script	Triggers when the browser is fetching the media data
onratechange	script	Triggers when the media data's playing rate has changed
onreadystatechange	script	Triggers when the ready-state changes
onredo	script	Triggers when the document performs a redo
onresize	script	Triggers when the window is resized
onscroll	script	Triggers when an element's scrollbar is being scrolled

onseeked	script	Triggers when a media element's seeking attribute is no longer true, and the seeking has ended
onseeking	script	Triggers when a media element's seeking attribute is true, and the seeking has begun
onselect	script	Triggers when an element is selected
onstalled	script	Triggers when there is an error in fetching media data
onstorage	script	Triggers when a document loads
onsubmit	script	Triggers when a form is submitted
onsuspend	script	Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched
ontimeupdate	script	Triggers when media changes its playing position
onundo	script	Triggers when a document performs an undo
onunload	script	Triggers when the user leaves the document
onvolumechange	script	Triggers when media changes the volume, also when volume is set to "mute"
onwaiting	script	Triggers when media has stopped playing, but is expected to resume

Chapter-5

What is jQuery?
jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto − Write less, do more. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code.
Here is the list of important core features supported by jQuery :

- DOM manipulation − The jQuery made it easy to select DOM elements, traverse them and modifying their content by using cross-browser open source selector engine called Sizzle.
- Event handling − The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
- AJAX Support − The jQuery helps you a lot to develop a responsive and feature-rich site using AJAX technology.
- Animations − The jQuery comes with plenty of built-in animation effects which you can use in your websites.
- Lightweight − The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
- Cross Browser Support − The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- Latest Technology − The jQuery supports CSS3 selectors and basic XPath syntax.

How to use jQuery?
There are two ways to use jQuery.

Local Installation − You can download jQuery library on your local machine and include it in your HTML code.

CDN Based Version − You can include jQuery library into your HTML code directly from Content Delivery Network (CDN).

Local Installation
Go to the https://jquery.com/download/ to download the latest version available.

Now put downloaded jquery-2.1.3.min.js file in a directory of your website, e.g. /jquery.

```
<html>

  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"  src = "/jquery/jquery-2.1.3.min.js"></script>

    <script type = "text/javascript">
      $(document).ready(function(){
        document.write("Hello, World!");
      });
    </script>
  </head>
  <body>
    <h1>Hello</h1>
  </body>

</html>
```

Plug-in: A plug-in is piece of code written in a standard JavaScript file. These files provide useful jQuery methods which can be used along with jQuery library methods.

How to use Plugins
To make a plug-in's methods available to us, we include plug-in file very similar to jQuery library file in the <head> of the document. We must ensure that it appears after the main jQuery source file, and before our custom JavaScript code.

Following example shows how to include jquery.plug-in.js plugin −

```
<html>
  <head>
    <title>The jQuery Example</title>

    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
    <script src = "jquery.plug-in.js" type = "text/javascript"></script>
    <script src = "custom.js" type = "text/javascript"></script>
    <script type = "text/javascript" language = "javascript">

      $(document).ready(function() {
        .......your custom code.....
      });
    </script>

  </head>

  <body>
    .............................
  </body>

</html>
```

In the above program create these two files(jquery.plug-in.js,custom.js) include them in the program . this is just a sample prog for demonstration purpose of using existing plugins.

How to develop a Plug-in
This is very simple to write your own plug-in. Following is the syntax to create a a method −

jQuery.fn.methodName = methodDefinition;
Here methodNameM is the name of new method and methodDefinition is actual method definition.

The guideline recommended by the jQuery team is as follows −

- Any methods or functions you attach must have a semicolon (;) at the end.
- Your method must return the jQuery object, unless explicity noted otherwise.

- You should use this.each to iterate over the current set of matched elements - it produces clean and compatible code that way.
- Prefix the filename with jquery, follow that with the name of the plugin and conclude with .js.
- Always attach the plugin to jQuery directly instead of $, so users can use a custom alias via noConflict() method.

For example, if we write a plugin that we want to name debug, our JavaScript filename for this plugin is −

jquery.debug.js
The use of the jquery. prefix eliminates any possible name collisions with files intended for use with other libraries.

Example
Following is a small plug-in to have warning method for debugging purpose. Keep this code in jquery.debug.js file −

```
jQuery.fn.warning = function() {
   return this.each(function() {
      alert('Tag Name:"' + $(this).prop("tagName") + '".');
   });
};
```

Here is the example showing usage of warning() method. Assuming we put jquery.debug.js file in same directory of html page.

```
<html>
   <head>
      <title>The jQuery Example</title>

      <script type = "text/javascript"
         src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script src = "jquery.debug.js" type = "text/javascript"></script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").warning();
            $("p").warning();
         });
      </script>
   </head>

   <body>
      <p>This is paragraph</p>
      <div>This is division</div>
   </body>

</html>
```

JSON Fundamentals

JSON or JavaScript Object Notation is a lightweight text - based open standard designed for human readable data interchange. Conventions used by JSON are known to programmers , which include C, C++, Java, Python, Perl ,etc.

- JSON stands for JavaScript Object Notation.
- The format was specified by Douglas Crockford.
- It was designed for human -readable data interchange.
- It has been extended from the JavaScript scripting language.
- The filename extension is .json.
- JSON Internet Media type is application/json.
- The Uniform Type Identifier is public.js on.

Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

Characteristics of JSON

- JSON is easy to read and write.
- It is a lightweight text-based interchange format.
- JSON is language independent.

Simple Example in JSON

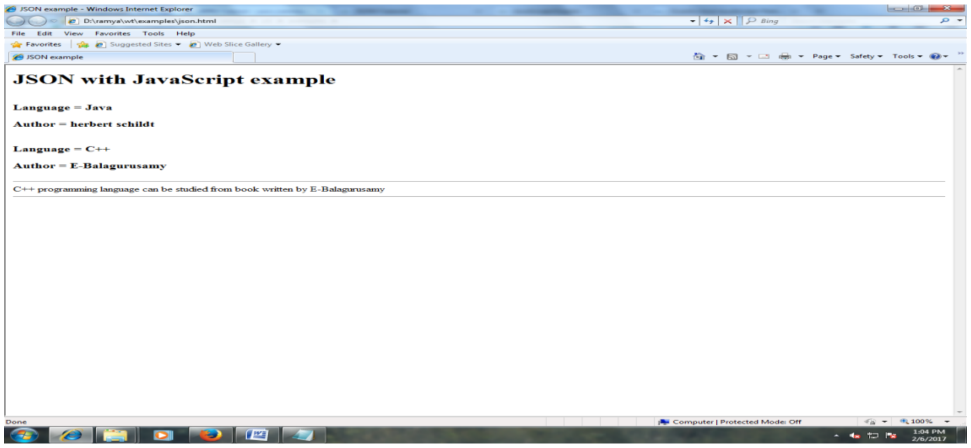
The following example shows how to use JSON to store information related to books based on their topic and edition.

```
{
"book":[
        {"id":"01",
        "language":"Java",
        "edition":"third",
        "author":"Herbert Schildt"
        },
        {
        "id":"07",
        "language":"C++",
        "edition":"second"
        "author":"E.Balagurusamy"
}]
}
```

After understanding the above program,we will try another example.Let's save the below code as json.htm

```
<html>
<head>
<title>
        JSON example
</title>
<script language="javascript">
var object1={"language":"Java",
                "author":"herbert schildt"
                };
document.write("<h1>JSON with JavaScript example</h1>");
document.write("<br>");
document.write("<h3>Language = "+object1.language+"</h3>");
document.write("<h3>Author = "+object1.author+"</h3>");
var object2 ={ "language":"C++",
                    "author":"E-Balagurusamy" };
document.write("<br>");
document.write("<h3>Language = "+object2.language+"</h3>");
document.write("<h3>Author = "+object2.author+"</h3>");

document.write("<hr />");
document.write(object2.language +" programming language can be studied "+"from book written by
"+object2.author);
document.write("<hr />");
</script>
</head>
<body>
</body>
</html>
```


JSON example - Windows Internet Explorer
D:\ramya\wt\examples\json.html
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
JSON example
Page Safety Tools

```
JSON with JavaScript example

Language = Java
Author = herbert schildt

Language = C++
Author = E-Balagurusamy

C++ programming language can be studied from book written by E-Balagurusamy
```

JSON syntax is basically considered as a subset of JavaScript syntax; it includes the following:
• Data is represented in name/value pairs.
• Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , comma).
• Square brackets hold arrays and values are separated by ,(comma).

JSON supports the following two data structures:
• Collection of name/value pairs: This Data Structure is supported by different programming languages.
• Ordered list of values: It includes array, list, vector or sequence etc.

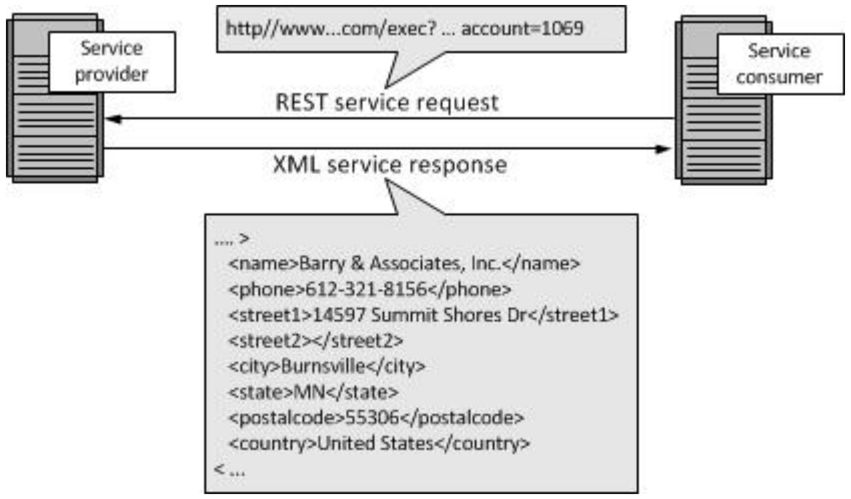
<u>JSON Value</u>
It includes:
• number (integer or floating point)
• string
• boolean
• array
• object
• null
Syntax
String |Number|Object|Array|TRUE |FALSE |NULL

REST style Services:

REST stands for **RE**presentational **S**tate **T**ransfer. REST is a web standards based architecture and uses HTTP Protocol for data communication. It revolves around resources where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in year 2000.

In REST architecture, a REST Server simply provides access to resources and the REST client accesses and presents the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource like Text, JSON and XML. JSON is now the most popular format being used in Web Services.

The following figure illustrates using REST for Web Services.



```
.... >
  <name>Barry & Associates, Inc.</name>
  <phone>612-321-8156</phone>
  <street1>14597 Summit Shores Dr</street1>
  <street2></street2>
  <city>Burnsville</city>
  <state>MN</state>
  <postalcode>55306</postalcode>
  <country>United States</country>
< ...
```

An application or architecture considered RESTful or REST-style is characterized by:

- State and functionality are divided into distributed resources
- Every resource is uniquely addressable using a uniform and minimal set of commands (typically using HTTP commands of GET, POST, PUT, or DELETE over the Internet)
- The protocol is client/server, stateless, layered, and supports caching

This is essentially the architecture of the Internet and helps to explain the popularity and ease-of-use for REST.