HTML5&CSS3

HTML BASICS

What is HTML?
HTML is the "mother tongue" of your browser.

To make a long story short, HTML was invented in 1990 by a scientist called Tim Berners-Lee. The purpose was to make it easier for scientists at different universities to gain access to each other's research documents. The project became a bigger success than Tim Berners-Lee had ever imagined. By inventing HTML he laid the foundation for the web as we know it today.

HTML is a language, which makes it possible to present information (e.g. scientific research) on the Internet. What you see when you view a page on the Internet is your browser's interpretation of HTML. To see the HTML code of a page on the Internet, simply click "View" in the top menu of your browser and choose "Source".

HTML is an abbreviation of "HyperText Mark-up Language" - which is already more than you need to know at this stage. However, for the sake of good order, let us explain in greater detail.

Hyper is the opposite of linear. In the good old days - when a mouse was something the cat chased - computer programs ran linearly: when the program had executed one action it went to the next line and after that, the next line and so on. But HTML is different - you can go wherever you want and whenever you want.

Text is self-explanatory.

Mark-up is what you do with the text. You are marking up the text the same way you do in a text editing program with headings, bullets and bold text and so on.

Language is what HTML is. It uses many English words.

Introduction to HTML:
Hyper Text Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. It is written in the form of HTML elements consisting of tags enclosed in angle brackets

HTML is a markup language for describing web documents (web pages). HTML stands for Hyper Text Markup Language. A markup language is a set of markup tags. HTML documents are described by HTML tags. Each HTML tag describes different document content.

HTML5&CSS3

Web browsers can read HTML files and render them into visible or audible web pages. Browsers do not display the HTML tags and scripts, but use them to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items.

HTML Files
Every web page is actually a HTML file. Each HTML file is just a plain-text file, but with a .html file extension instead of .txt, and is made up of many HTML tags as well as the content for a web page.

A short history of HTML
The founder of HTML was Tim Berners-Lee and his product was made attractive to the general public by Mosaic browser which was evolved at NCSA. It has become extremely popular and well-known in the 1990's when the Internet had been developing rapidly.

The vision of the HTML developers is that all devices must be able to reach the data on the Internet: computers with different platforms, browsers and characteristics, pocket devices, cell phones, devices for speech, and many others.

About HTML:
1. HTML is used to create webpages.
2. HTML used many tags to make a webpage. So it is a tag based language.
3. The tags of HTML are surrounded by angular bracket.
4. It can use wide ranges of colors, objects and layouts.
5. Very useful for beginners in web designing field.

What are the advantages of HTML?
The HTML is the widely used language for writing web pages. HTML is highly flexible and user friendly. HTML is similar to that of XML, which is used in data storage. HTML has many advantages as shown below:

1. HTML is highly flexible
2. HTML is supported on almost every browser
3. HTML is user friendly
4. HTML is an open technology
5. HTML is consistent and efficient
6. HTML bids the superlative medium for its users
7. HTML is easily understandable and does not require any training

8. HTML is designed with a feature of interaction between the webpages, which makes it effective
9. HTML provides search engine compatible pages
10. HTML is easier to maintain and update any site
11. HTML does not involve strain on the servers
12. For HTML web pages, it takes less time to load the web pages
13. HTML validation is another important key factor which increases the web accessibility
14. HTML webpage's look and feel attracts larger masses to visit the website

Disadvantages of HTML:
1. It can create only static and plain pages so if we need dynamic pages then HTML is not useful.
2. Need to write lot of code for making simple webpage.
3. Security features are not good in HTML.
4. If we need to write long code for making a webpage then it produces some complexity.

What are the different versions of HTML?
HTML is an evolving language. HTML elements or tags are the building blocks for web pages. In the year 1991, "HTML tags" was the first document that came into existence on internet by Berners-Lee. Initially, HTML was considered as an application of SGML. The following are the different versions of HTML:

HTML 1.0 (1989 - 1994)
The first version of HTML that supported inline images and text controls. HTML 1.0 was very limited in terms of styling and presentation of content. You could not: use tables or frames, specify fonts, change page background, and use forms

HTML 2.0 (1995)
This specification supported more browsers. HTML 2.0 was considerably improved to support: It also supported:
1 forms with limited set of form elements such as text boxes, and option buttons
2 change of page background
3 uses of tables

HTML 3.20 (1997)
After HTML 2.0, the next major version development was HTML 3 which happened in late 1995. But HTML 3 development did not get completed and it was never implemented. The next major release after HTML 2.0 was HTML 3.2, a version release by W3c. The HTML 2.0's IETF ended its HTML Working Group in September 1996. HTML 3.2 was published in January 1997. HTML 3.2 included tables; attribute alignment, images, headings and few more.

HTML 4.01 (1999)

HTML5&CSS3

It is the current official version of HTML. HTML 4.0.1 enhanced most trademarked extensions. In December 1999, HTML 4.0.1 came into existence with support for internationalized documents, CSS, added table forms.

HTML 5
It is the latest version of HTML. HTML 5 came into life on January 2008. HTML 5 included its own HTML serialization along with XML based XHTML 5 serialization. HTML 5's syntax brings to mind SGML syntax.

What is TAG/Element?
HTML tags are the hidden keywords within a web page that define how the browser must format and display the content.

Syntax:	<------>
Example:	<html>

Types of Tags/Elements:
Most tags must have two parts, an opening and a closing part. For example, <html> is the opening tag and </html> is the closing tag. Note that the closing tag has the same text as the opening tag, but has an additional forward-slash (/) character. I tend to interpret this as the "end" or "close" character.

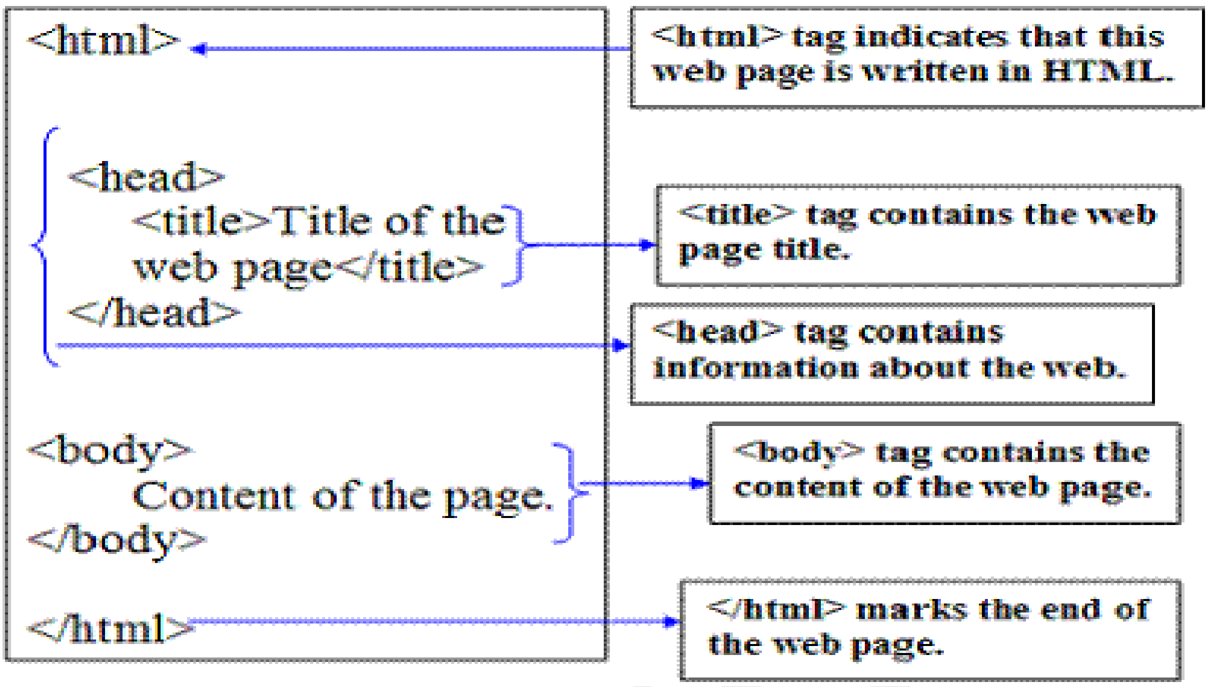
1.) Paired tags:
The tags that have both opening and closing tags are called as Paired tags
Examples:
<title>-----------------</title>
<head>-----------------</head>

Note: The closing tag starts with a forward slash ("/")

2.) Non-Paired tags
The tags that have only opening tags but no closing tags are called as non-paired tags.

Examples:
<br> (break)
<hr> (Horizontal-Rule)
<img> (Image)

Structure of html:


<html>
<head>
<title>Title of the web page</title>
</head>
<body>
Content of the page.
</body>
</html>
<html> tag indicates that this web page is written in HTML.
<title> tag contains the web page title.
<head> tag contains information about the web.
<body> tag contains the content of the web page.
</html> marks the end of the web page.

Essential HTML Tags
There are four sets of HTML tags that form the basic structure needed for every HTML file:

1 <html></html>
2 <head></head>
3 <title></title>
4 <body></body>

Definition - <html> </html>
This basically defines the document as web page. It also identifies the beginning and end of the HTML document. All other tags must fall between the html tags.

Header - <head> </head>
The header contains information about the document that will not appear on the actual page, such as the title of the document, the author, which style sheet to use and also meta tags.

Title - <title> </title>
The title tag defines the title that will appear in the title bar of your web browser. The title must appear between the head tags.

Body - <body> </body>
The body tags contain all the information and other visible content on the page. All your images, links and plain text must go between the <body> and </body> tags.

HTML5&CSS3

HTML editors
There are so many software packages available to develop HTML. The software packages can be grouped into two main categories:

1 text-based (or code-based) Editors
2 WYSIWYG (what you see is what you get) Editors

Text-based (or code-based) editors
To start creating web pages, you do not need an expensive software package but you do need some knowledge of HTML. You can create web pages with a basic text editor like Windows Notepad. However, as you master scripting HTML, you will learn that using Notepad or some other basic text editor is not sufficient. In Windows, Notepad can be started from the Start Menu:

WYSIWYG editors
Because WYSIWYG HTML editors do not require much HTML knowledge, they tend to be expensive. These editors allow you to directly work in the "design" or "preview" view instead of the code view. The main advantage of working with the design view is that you can design the layout of your page by dragging-and-dropping pieces of your page layout. Thus a web page can be developed more quickly than by hard-coding it by hand using a text-based editor.

There are several popular WYSIWYG editors available:
1 Macromedia Dreamweaver
2 Microsoft FrontPage
3 Adobe Go Live

Text-based editors Vs. WYSIWYG editors

Text-based (or code-based) Editors	WYSIWYG Editors
Better control. Because text-based editors require knowledge of HTML, the developer has more control over what is written to produce a web page. In some cases, a software package (like FrontPage) may write proprietary code that may not be interpreted by all the browsers. *Faster editing*. You can quickly change your code unlike WYSIWYG editors.	*Support for other scripting languages.* WYSIWYG editors provide advanced features to code in other programming/scripting languages such as JavaScript, XHTML, ASP/ASP.NET, PHP, and JSP. *Faster/simplified development*. WYSIWYG editors allow you to develop pages quickly as the software writes the necessary code in response to the layout you design for your

WYSIWYG editors require, for example, a lot of computer-resources to start-up or load/open a page. *More flexibility*. You can edit your code directly at the desired location. This cannot be always done with WYSIWYG editors.	web page. *Better organization*. Dreamweaver, for instance, allows you to define a site folder that contains all the files that make up the website. By defining a local site folder, you have many advantages including moving of files (without breaking links), searching of a particular term or tag in the entire site (without having the file open!), and FTP support to move only changed or new files to the server.

Creating an HTML document
Before you start writing code to write a web page, it is a good practice to plan ahead the appearance of the web page. An HTML document has two elements:

1 Document Content
2 Tags or Elements

Naming conventions
HTML files names are very important as they are used to locate or open the files. Remember these points when naming HTML files:

1 Save your web page files with the .htm or .html file extension. (Both of these files represent HTML files, older systems such Windows 3.1 and DOS cannot recognize four-letter file extensions. Because the first three letters of .html and .htm are the same, those systems ignore the "l" and just recognize ".htm".)

2 Some web servers are case-sensitive. That means those web servers would consider page1.htm and Page1.htm as two different files. To avoid case sensitivity problems, use only lowercase or uppercase letters to name your files.

3 Filenames should consist only of letters and numbers. Avoid using spaces, punctuation, or special characters. If you need to separate words, use dashes (-) and underscores (_), for example, creating-an-HTML-document.htm or creating_an_HTML_document.htm.

Understanding elements
To markup your web pages, you will need to learn the syntax (rules of a computing language) of HTML. HTML syntax is very simple to follow. Remember the syntax only controls the presentation or structure of a web page. The most fundamental piece of building block of HTML is the elements.

HTML5&CSS3

An Element refers to two different things:
1 Element of structure of a document (for example, paragraph, web page title, etc.).
2 Element in the sense of a tag (for example, <p>, <title>)

Main points to remember:
1 Every element has a name such as head, title, p, i, and b.

2 A tag is the element name surrounded by the angled brackets. This refers to a start tag such as <p>, <title>, and <i>. A start tag starts a particular HTML instruction.

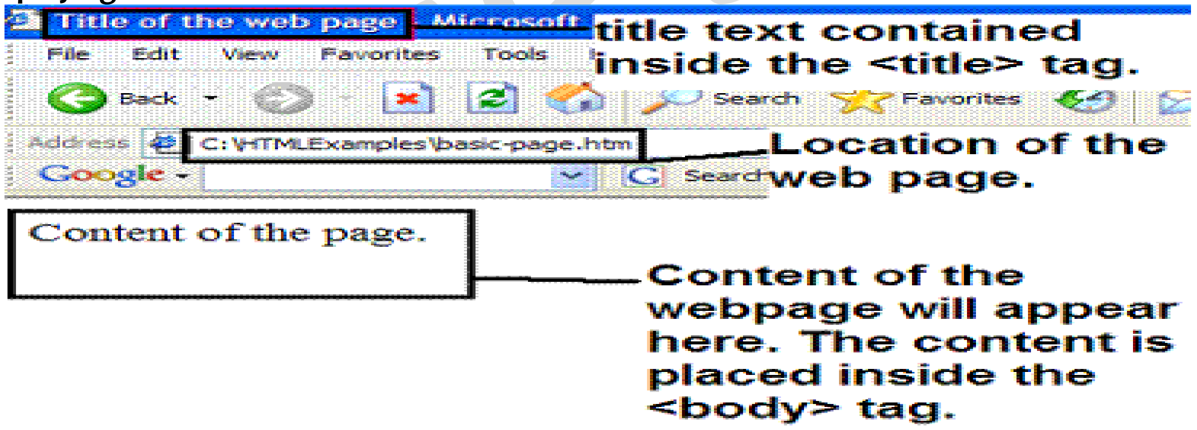
3 An end tag is the same as a start tag except the end tag has a forward slash between the < and the element name. An end tag stops a particular HTML instruction.

4 Most elements have content, which is placed between the start and the end tags. Example, this is <b>bold</b>.

5 Some elements have no content. Such elements/tags are known as empty tags.

6 Some elements have no end tags. These are referred to as one-sided tags. A tag that has an opening and closing tag is referred to as two-sided tag.

Displaying an HTML document



title text contained inside the <title> tag.

Location of the web page.

Content of the webpage will appear here. The content is placed inside the <body> tag.

HTML5 Comment Tag
As you create more pages, it may become difficult for you to understand your code in the future unless you use comments. By adding comments to your code, it could be a valuable reference guide for you when you look at your code in the future. Ideally, comments should be added to indicate what the code does. Comments are personal; so add anything that you think is impotent about your code.

In HTML, a comment **begins with <! -- And ends with -->**. Any text you place after <!-- is comment. Browsers ignore comment text. Again, comment is for your reference; it does not get displayed on the web page. In a comment, you can freely include special characters, such as ampersands, quotation marks, and angle brackets. Your comments can also span multiple lines. The browser will stop ignoring text once it reads -->.

Example of a single line comment:
<!--This is a small comment-->

Example of a multi-line comment:
<!--This comment is long. It is displayed on more than one line. Adding multi-line comments in HTML is easy as adding a single line comment. Whether the comment is single line or multi-line, it starts with <-- and ends with -->

Important points about comments:

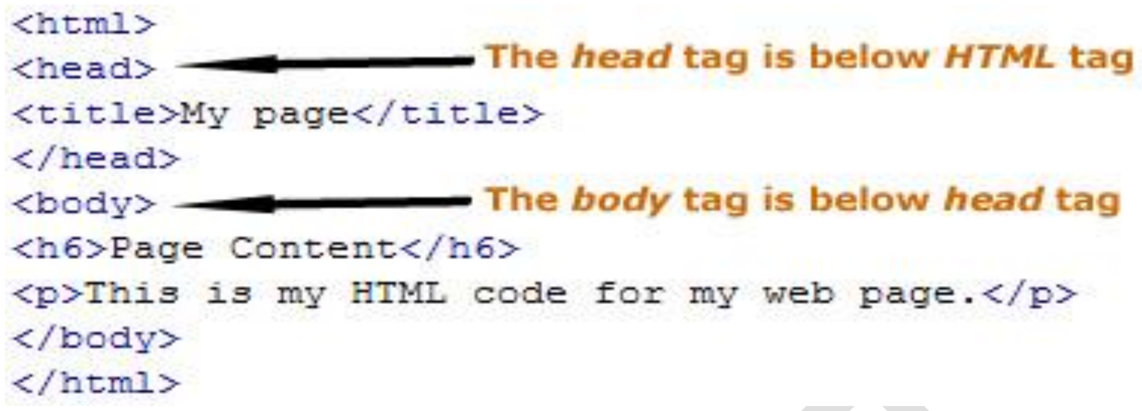
- ✓ Use comments as a reference guide. Avoid use of excessive comments.
- ✓ Add comments to major parts of your code; or for parts, that you are unsure of.
- ✓ Use of comments is a great way to communicate with other people working on the same web page.
- ✓ Start your comment with <-- and end with -->
- ✓ Browsers and search engines ignore comments.

Spacing and breaks:
The amount of space you use for your HTML code can really affect how quickly and easily you understand your code. When your HTML document is long because it contains many tags, it becomes necessary to use space and breaks to identify where a tag begins and ends. Without the use of right amount of space in your code, it becomes difficult to manage HTML code.

If you use space more than needed, then, you may have to scroll more to follow your code. Although there are no hard rules about how much space you can and cannot use, the following identifies areas of HTML code for proper amount of spacing:

1 breaks between tags
2 spacing between tags
3 spacing inside the brackets

Breaks between tags
The first place where you can add breaks is to the following basic tags: <html>, <head>, <title> </title>, </head> <body>, </body> and </html>. The following shows an example of breaks between these basic and other tags:



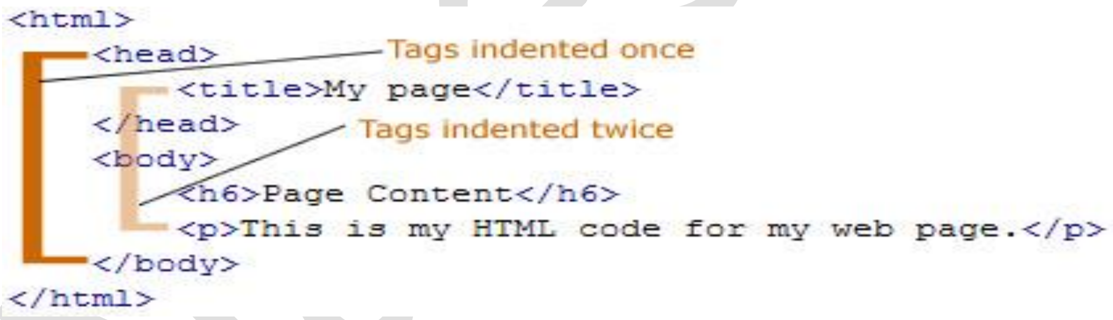
```
<html>
<head>
<title>My page</title>
</head>
<body>
<h6>Page Content</h6>
<p>This is my HTML code for my web page.</p>
</body>
</html>
```

The *head* tag is below *HTML* tag
The *body* tag is below *head* tag

Note above each tag is on its own line. (This does not mean that each tag should be placed on its own line. If, for instance, your tag affects only a few words (i.e., link or bold text) within a paragraph, you may not need to break the line at each individual tags.) Simply use the RETURN or ENTER key to add breaks between tags.

Spacing between tags

In addition to adding breaks, you can also add space (or tabs) to enhance the understanding of your code. Because the <html> tag is the root tag of an HTML document, you may not need to use any extra spacing for this tag. All other tags, however, inside of this root tag should be indented:

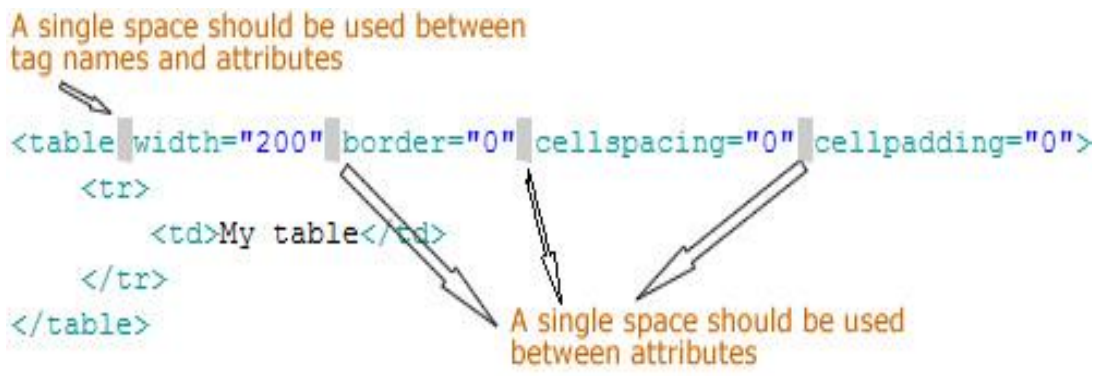


```
<html>
    <head>
        <title>My page</title>
    </head>
    <body>
        <h6>Page Content</h6>
        <p>This is my HTML code for my web page.</p>
    </body>
</html>
```

As indicated above, the <head>, and <body> tags and their corresponding closing tags are indented once. The inner tags (<title>, <h6>, and <p>) inside these main tags are indented twice.

Spacing inside the brackets

Lastly, you can add space inside the angle brackets of a tag to enhance the understanding of your code. You can add space inside the angle brackets when you use attributes:

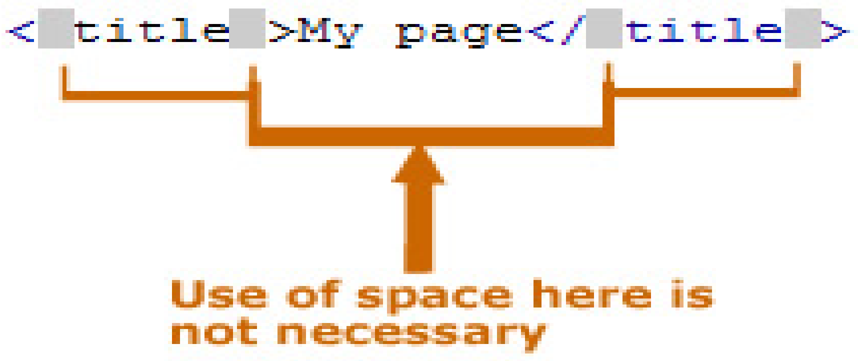

A single space should be used between tag names and attributes
A single space should be used between attributes

```
<table width="200" border="0" cellspacing="0" cellpadding="0">
    <tr>
        <td>My table</td>
    </tr>
</table>
```

Avoiding unnecessary space in HTML code
As you try to help yourself to make your code more readable, you may start using space where it is not needed. Remember browsers ignore any extra space you may have in your code. That does not mean you should use extra space when it is not needed, however. There are two areas where the use of space should be avoided:

1 Space between the tag name and brackets
2 Space between the tag name and the text it affects

There should be no space between a tag name and its angle brackets. (This page explains when to use space inside the brackets when you are using attributes.) As an example, avoid.

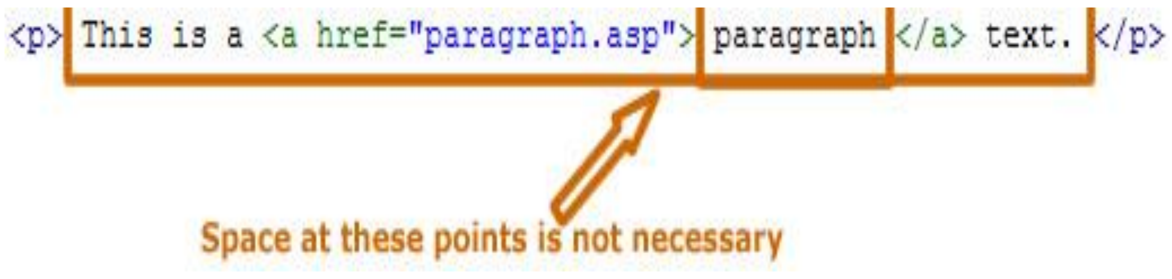


```
< title >My page</ title >
```

There is no reason to use space between a tag name and the angle brackets. So the title tag above should be rewritten as:

<title>my page</title>

Space between the tag name and the text it affects
Space should also be avoided when you use a tag that affects some text (or some other web object):



```
<p> This is a <a href="paragraph.asp"> paragraph </a> text. </p>
```

Parts in HTML Document:
Generally HTML document has the following 3 parts:

1. A line containing HTML version information,
2. A declarative header section (delimited by the HEAD element),
3. A body, which contains the document's actual content.

1. HTML version information (HTML <! DOCTYPE> Declaration)
The <! DOCTYPE> declaration must be the very first thing in your HTML document, before the <html> tag. The <! DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in. A document type declaration, or DOCTYPE, is an instruction that associates a particular SGML or XML document (for example, a webpage) with a document type definition (DTD)

This is the first line of code of any HTML page and it should be always in first place. This short code asks the browser to render your page using HTML5 markup language. There is no version number mentioned in the code which simply refers to the current version.

Definition and Usage
The <! DOCTYPE> declaration must be the very first thing in your HTML document, before the <html> tag.

The <! DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in.

In HTML 4.01, the <! DOCTYPE> declaration refers to a DTD, because HTML 4.01 was based on SGML. The DTD specifies the rules for the markup language, so that the browsers render the content correctly.

HTML5 is not based on SGML, and therefore does not require a reference to a DTD.

COMMON DOCTYPE DECLARATIONS

HTML 4.01 Strict

This DTD contains all HTML elements and attributes, but does NOT INCLUDE presentational or deprecated elements (like font). Framesets are not allowed.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

HTML 4.01 Transitional
This DTD contains all HTML elements and attributes, INCLUDING presentational and deprecated elements (like font). Framesets are not allowed.

<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

HTML 4.01 Frameset
This DTD is equal to HTML 4.01 Transitional, but allows the use of frameset content.

<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

XHTML 1.0 Strict
This DTD contains all HTML elements and attributes, but does NOT INCLUDE presentational or deprecated elements (like font). Framesets are not allowed. The markup must also be written as well-formed XML.

<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

XHTML 1.0 Transitional
This DTD contains all HTML elements and attributes, INCLUDING presentational and deprecated elements (like font). Framesets are not allowed. The markup must also be written as well-formed XML.

<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

XHTML 1.0 Frameset
This DTD is equal to XHTML 1.0 Transitional, but allows the use of frameset content.

<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

XHTML 1.1

This DTD is equal to XHTML 1.0 Strict, but allows you to add modules (for example to provide ruby support for East-Asian languages).

```
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

HTML 5
HTML5 is not based on SGML, and therefore does not require a reference to a DTD.

Example:
HTML5 Example:

```
<! DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>
<body>
Content of the document......
</body>
</html>
```

Meta Charset:

```
<meta charset="utf-8" />
```

Charset=UTF-8 stands for Character Set = Unicode Transformation Format-8. It is an octet (8-bit) lossless encoding of Unicode characters. UTF-8 is a universal encoding that will support most of the character like Greece, Hebrew, Japanese, etc.

Character encoding
HTML-4.0

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

HTML-5.0:

```
<meta charset="utf-8">
```

What is Unicode?
Unicode provides a unique number for every character,

- ✓ no matter what the platform,
- ✓ no matter what the program,
- ✓ no matter what the language.

The byte-order mark (BOM) in HTML

At the beginning of a page that uses a Unicode character encoding you may find some bytes that represent the Unicode code point U+FEFF BYTE ORDER MARK (abbreviated as BOM).

Where is a BOM useful?
A: A BOM is useful at the beginning of files that are typed as text, but for which it is not known whether they are in big or little endian format.

Specify Character Set
You can use <meta> tag to specify character set used within the webpage.

Example
By default, Web servers and Web browsers use ISO-8859-1 (Latin1) encoding to process Web pages. Following is an example to set UTF-8 encoding:

Example1:
```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="author" content="ksNareshIT" />
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<p>Hello HTML5</p>
</body>
</html>
```
To serve the static page with traditional Chinese characters, the webpage must contain a <meta> tag to set encoding:

Example2:
```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="HTML, Meta Tags, Metadata, good HTML5 trainer" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="author" content="ksNareshIT" />
<meta http-equiv="Content-Type" content="text/html; charset=Big5" />
</head>
<body>
<p>Hello HTML5</p>
```

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

Lang Attribute:
The HTML Lang attribute can be used to declare the language of a Web page or a portion of a Web page. This is meant to assist search engines and browsers. The html element is the root of HTML document; the whole document resides within the paired tags. It is recommended to declare the primary language of your webpage with its Lang attribute. Its value is a two letter code which represents a language.

Alternatively, you can also put a two letter country code in the value separated by a hyphen (-). The code represents a country where the language is spoken.

1. English (U.S.)
```
   <html lang="en-US">
   ...
   </html>
```

2. English (U.K./Great Britain)
```
   <html lang="en-GB">
   ...
   </html>
```

3. English(IN)
```
<html lang="en-IN">
...
</html>
```

Head Element:
It contains general information, meta-information and document type information. The Head Tag inside Elements should not display in body part in web browser.

HTML head Elements

TAG	**DESCRIPTION**
`<title>`	Defines the title of a document
`<link>`	Defines the relationship between a document and an external resource
`<meta>`	Defines metadata about an HTML document
`<script>`	Defines a client-side script
`<style>`	Defines style information for a document

The HTML `<title>` Element

The <title> tag defines the title of the document. The <title> element is required in all HTML/XHTML documents. It is a paired tag.

Syntax:
<title>....................</title>

Example:
<title>Welcome to NareshIT - Hyderabad. Register Now for New Batches - Naresh i Technologies</title>

The HTML <link> Element
The <link> tag defines the relationship between a document and an external resource.
The <link> tag is most used to link to style sheets. It is a non-paired tag.

Syntax:
<link>.

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

How to Add a Favicon to your Site
Favicons (also called shortcut icons) first appeared in Internet Explorer 5, where placing a favicon.ico icon in the root of a website would cause a 16px square image to appear next to the URL

A favicon is a graphic image (icon) associated with a particular Web page and/or Web site. To add a favicon to your Web site. The format of the image must be one of PNG (a W3C standard), GIF (Graphic Interchange Format) , or ICO.

The format for the image you have chosen must be 16x16 pixels or 32x32 pixels, using either 8-bit or 24-bit colors. The format of the image must be one of PNG (a W3C standard), GIF, or ICO.

What sizes are needed?
If you were to create a favicon for every possible use, the sizes you would need to create are:
- ✓ 16px: For general use in all browsers, could be displayed in the address bar, tabs or bookmarks views!
- ✓ 24px: Pinned Site in Internet Explorer 9
- ✓ 32px: New tab page in Internet Explorer, taskbar button in Windows 7+ and Safari's 'Read Later' sidebar

- ✓ 57px: Standard iOS home screen (iPod Touch, iPhone first generation to 3G)
- ✓ 72px: iPad home screen icon
- ✓ 96px: Favicon used by the Google TV platform
- ✓ 114px: iPhone 4+ home screen icon (twice the standard size for the retina display)
- ✓ 128px: Chrome Web Store
- ✓ 195px: Opera Speed Dial

A favicon (short for "favorites icon"), also known as a page icon or an urlicon, is an icon associated with a particular website or webpage. A web designer can create such an icon, and many recent web browsers can then make use of them. Browsers that support favicons may display them in the browser's URL bar, next to the site's name in lists of bookmarks, and next to the page's title in a tabbed document interface.

Example:

```
<html>
<head>
<title>
MyIcon
</title>
<link href="circle.ico" rel="icon" type="image/ico" >
</head>
</html>
```

Example:

```
<html>
<head>
<title>
My-Library
</title>
<link href="book-library.png" rel="icon" type="image/ico" >
</head>
</html>
```

Specifying Metadata

The <meta> tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable. Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata

Metadata helps search engines track and index web pages. There are number of meta tags that need to be included in any web page for it to be indexed by search engines. Search engines are influenced by the metadata you provide in your web pages using meta tags.

Why should we specify any data in our web pages for search engines? You may be thinking we write the pages for people but not for search engines. Actually, a successful website cannot afford to exclude information that helps search engines to index pages. Search engines bring traffic to websites. (Think of the last time you used a major search engine such as www.google.com, www.yahoo.com, or www.msn.com)

Search engines use meta tags and the content of the web page to determine where your page is displayed when someone makes a search in a particular search engine. Obviously, you want your pages to come up first or near the top when a search engine returns results. To give you an example, if you go to Google.com and type nareshit in the search box, chances are that www.nareshit.com will come up first or on the first page even though Google displays thousands of other related pages to "nareshit" So the idea is that a web searcher is more likely to click on a link that comes up first or near first than a page that is more clicks away.

Meta tag attributes
Meta tags are used to provide information about a web page. The meta tags are optional and easy to use. There are three main attributes of meta tags: name, content, and http-equiv. The name attribute specifies the type of information. The content attribute includes the meta-information. (Access this page to learn how to use these two attributes to optimize your web page for search engines.) Lastly, the http-equiv attribute specifies a particular header type.

Name attributes
The purpose of using name attributes is to provide information to the search engines. The following table summarizes the most commonly use meta name attributes:

Attribute name	**Explanation**
description	Description of the web page.
keywords	Used to list keywords those describe the content of the web page.
creator or author	The organization or person who responsible for creating the webpage.
date	The date of publication in yyyy-mm-dd format.
identifier	A unique number identifying a web page.
language	Language of the page. Use a two-character
rights	Used for adding copyright statement.

Http-equiv attributes

The http-equiv attributes are equivalent to HTTP headers and control the action of a browser on a requested page. When displaying, the browser uses the instructions specified by the http-equiv attributes. The instructions may contain information about when the content of the page will expire or when the page should be refreshed.

The HTML <meta> Element
The <meta> tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable. Meta elements are typically used to specify page description, keywords, author of the document, last modified and other metadata.

The metadata can be used by browsers (how to display content or reload page), search engines (keywords), or other web services. <meta> tags always goes inside the <head> element.

Meta Keywords:
Define keywords for search engines:
Syntax:
<meta name="keywords" content="required list of keywords">

Example:
<meta name="keywords" content="java beginner, beginner java tutorial, java online tutorial, java source code"/>

Example:
<meta name="keywords" lang="en-IN" content="live cricket scores, cricket, india cricket, live cricket news, yahoo cricket, australia cricket, england cricket"/>

Meta Description
Define a description of your web page:

Syntax:
<meta name="description" content="required Page Description">

Example:
<meta name="description" content="A online beginner java tutorial website covering basics of programming along with java source code." />

Example:
<meta name="description" lang="en-IN" content="Check out live cricket scores, cricket news headlines, cricket schedules &amp; results and more from Yahoo!"/>

Meta Author

Define the author of a page:
Syntax:
```
<meta name="author" content="name of the author">
```

Example:
```
<meta name="author" content="ksNareshIT" />
```

Meta Title:
Define the title of the meta content.
Syntax:
```
<meta name="title" content="required meta title">
```

Example:
```
<meta name="title" content="Welcome to NareshIT - Hyderabad. Register Now for New Batches" />
```

Example with all Meta Contents:
```
<!DOCTYPE html>
<html lang="en-IN">
<head>
<title>
Related Meta Content and Information
</title>
<meta charset="UTF-8">
<meta name="description" content="best training institute">
<meta name="keywords" content="HTML, CSS,XML, JavaScript">
<meta name="author" content="ksNareshIT">
<meta name="title" content="Welcome to NareshIT - Hyderabad. Register Now for New Batches" />
</head>
<body>
<p>All meta information goes before the body.</p>
</body>
</html>
```

The HTML <style> Element
The <style> tag is used to define style information for an HTML document.
Inside the <style> element you specify how HTML elements should render in a browser:

Example:
```
<head>
<style type="text/css">
body
```

```
{
background-color:yellow;
}
</style>
</head>
```

The HTML <script> Element

The <script> tag is used to define a client-side script, such as a JavaScript.

Example:

```
<head>
<script type='text/javascript' language="javascript">
function MyDate()
{
document.getElementById("myd").innerHTML=Date();
}
</script>
</head>
<body>
<p id="myd">Click the button to display the Current System Date and Time...</p>
<button onclick="MyDate()">ClickMe</button>
</body>
```

BODY Section:

The body of the document contains all that can be seen when the user loads the page. In the rest of this tutorial you can learn in detail about all the different aspects of HTML, including:

1 Text
- Formatting
- Resizing
- Layout

2 Links
- To local pages
- To pages at other sites
- To bookmarks

3 Images
- Inserting images (GIF and jpg)
- Adding a link to an image

4 Backgrounds
- Colors
- Images

- Fixed Image

5 Tables
6 Frames
7 Forms
8 Hexadecimal Colors

The body element section all the contents of an HTML document, such as text, hyperlinks, images, Special Character, lists, tables, frames, forms etc. Mostly all the HTML Tags are use this Section. It's most powerful section. It is a paired tag.

Syntax:
<body>......................</body>

Body tag attributes and parameters:

Attributes	Parameters
bgcolor	Color Name/Color Code
background	image path
text	Color Name/Color Code

Note:
ColorCode Indicates Hexadecimal color code or number.

Example1:
```
<!DOCTYPE html>
<html lang="en-IN">
<head>
<title>
Working with body tag
</title>
</head>
<body>
Welcome to Body Section....
</body>
</html>
```

Example2:
```
<!DOCTYPE html>
<html lang="en-IN">
<head>
<title>
Working with body tag and Attrs.
</title>
```

```
</head>
<body bgcolor="yellow" text="blue" background="html5.png">
Welcome to Body Section....
</body>
</html>
```

Example3:

```
<!DOCTYPE html>
<html lang="en-IN">
<head>
<title>
Working with body tag and ColorCodes
</title>
</head>
<body bgcolor="#FFFF00" text="#FF00FF" background="D:\HTML Class Materials\HTML5\Images\html5.png">
Welcome to Body Section....
</body>
</html>
```

Example4
Example3:

```
<!DOCTYPE html>
<html lang="en-IN">
<head>
<title>
Working with body tag and ColorCodes
</title>
</head>
<body bgcolor="#FFFF00" text="#FF00FF" background="file:///D:\HTML Class Materials\HTML5\Images\html5.png">
Welcome to Body Section....
</body>
</html>
```

Understanding attributes

In HTML, elements (or tags) have attributes or properties. As an HTML writer, attributes allow you to add extra instruction to your tags. Because each tag has its own unique attributes, you have to learn which attribute(s) belongs which tag. (See the attributes reference table for details.) Any attribute cannot be just applied to any tag.

An attribute has two parts: attribute name and attribute value. Because of these two-parts, they are also referred to as pairs. The attribute name identifies (or defines) what special instruction you want to add to a particular tag. The attribute value, on the other hand, indicates (usually predefined) option for that attribute. So if you are going to use an attribute, you will need to have value for that attribute. Let's go over the actual HTML.

Align="right" is an example of attribute-value pair. The word align is the attribute. The value of this attribute is right. A value of an attribute is enclosed in double quotation marks. Notice the value is on the right-hand side of the equal sign and the attribute name is on the left of equal sign.

Keep the following points in mind while working with attributes:
1 Some attributes have predefined values. For example, for the align attribute, possible values include, left, center, justify and right. So if you use the align attribute, you should use one of these acceptable values.

2 Some attributes accept numerical values. For instance, for the width attribute, you can specify a numerical value such as 5 (which indicates 5 pixels), or 20% (which indicates 20% of the screen width).

Main points to remember for attributes:

1 Attributes are specific to tag names. For example, for the <p> tag, you can use the align attribute but not the width attribute. The width attribute can only be used with tags such as <table>, <td>, and <img>.

2 Attributes have values. Make sure to use the correct value for the correct attribute. For instance, you should not use color="20", or align="brown"; instead use, color="red", and align="justify".

3 Attribute values needs to be enclosed in double quotation marks. This is true especially if the value contains one or more spaces, for example, face="Times New Roman".

4 Attribute values could come from a predefined list (such as color names red, green, blue, etc.) or from you (width of a table 50% or 800 pixels.)

HTML Attributes and Parameters:
1. HTML tags can contain one or more attributes.
2. Attributes are always specified in the start tag.
3. Attributes consist on name/value pairs.
4. Attribute values always be enclosed in quotes.
5. Double quotes are the most common use, but single quotes are also allowed.
6. Attributes are special features of a tag.

Parameters:

Parameters are the values, that we assign to an attribute.

Syntax:
    <tag attribute="parameter">
Example:
    <body bgcolor="blue">

Different Types of Attributes:
1. Element-Specific Attributes (Element-Tag)
Example:
<body> --> bgcolor,background,text
<img> --> src,width,height,alt,align....

----

2. Global Attributes(Standard Attributes): These attributes are common for all elements.They are
1.class
2.id
3.lang
4.insert
5.spellcheck
6.style

Example:
```
<body background='html5.png'
style="background-repeat:no-repeat;background-attachment:fixed;text-align:justify">
Sometext
Sometext
Sometext
</body>
```

3. Event Handler Content Attributes: These are related to JavaScript Events and Event Handlers. They are the following:
1. onclick
2. oninput
3. onprogress
4. onchange
5. oninvalid
6. onLoad

Example:

```
<head>
<script type='text/javascript' language='javascript'>
function Welcome()
{
alert("Welcome to Event Atrributes");
}
</script>
</head>
<body>
<input type='button' onclick="Welcome()" value="Message"/>
</body>
```

Optional type attributes

HTML5 specifies default values for the type attribute on script, style and link elements. Unless you need a different value than the default, you can simply omit the type attribute.

HTML 4.01
```
<link rel="stylesheet" type="text/css" href="hi.css">
```
HTML5
```
<link rel="stylesheet" href="hi.css">
```

HTML 4.01
```
<script type="text/javascript" src="hello.js"></script>
```
HTML5
```
<script src="hello.js"></script>
```

HTML 4.01
```
<style type="text/css">
  body { background: gray url(boring.gif) no-repeat; }
</style>
```

HTML5
```
<style>
  body { background: pink url(unicorns.png) repeat; }
</style>
```

Optional solidus (/>)

The most distinct difference between HTML4 and XHTML is that the latter requires a stricter syntax for self-closing elements.

The space is optional, so the following is valid XHTML as well in HTML5

```
<img src="html5.png" alt="MyImage"/>
```

However, in HTML, using the solidus isn’t required:

```
<img src="html5.png" alt="MyImage">
```

Color Code System (HTML Colors)
W3C Standard 16 Colors:
Here is the list of W3C Standard 16 Colors names and it is recommended to use them.

Black	Gray	Silver	White
Yellow	Lime	Aqua	Fuchsia
Red	Green	Blue	Purple
Maroon	Olive	Navy	Teal

Color Values
HTML colors are defined using a hexadecimal notation (HEX) for the combination of Red, Green, and Blue color values (RGB). The lowest value that can be given to one of the light sources is 0 (**in HEX: 00**). The highest value is 255 **(in HEX: FF)**. HEX values are specified as 3 pairs of two-digit numbers, starting with a # sign.

HTML Colors - Hex Codes:
A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB). A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Advanced Paint Brush. Hexadecimal Value - #FFFFFF RGB Color Code - R=255 G=255 B=255

16 Million Different Colors
The combination of Red, Green, and Blue values from 0 to 255, gives more than 16 million different colors (256 x 256 x 256).

```
#FF0000-->Red
#00FF00-->Green
#0000FF-->Blue
#000000-->Black
#FFFFFF -->white
------------
------------
```

Example:

```
<!DOCTYPE html>
<html lang="en-IN">
<head>
<title>
```

```
Working with body tag and ColorCodes
</title>
</head>
<body bgcolor="#FFFF00" text="#FF00FF" background="D:\HTML Class Materials\HTML5\Images\html5.png">
Welcome to Body Section....
</body>
</html>
```

ENTITIES: (HTML SPECIAL CHARACTERS)

Character entities can be typed as either a numbered entity or a named entity. All character entities begin with an ampersand (&) and end with a semicolon (;). Although every character entity has a numbered version, not everyone has a named version. While a few are listed in the following table to give you an idea of what they look like.

A Few Character Entities:

Result	Description	Entity Name	Number Code
	non-breaking space	&nbsp;	&#160;
¡	inverted exclamation mark	&iexcl;	&#161;
¤	currency	&curren;	&#164;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
¦	broken vertical bar	&brvbar;	&#166;
§	section	&sect;	&#167;
¨	spacing diaeresis	&uml;	&#168;
©	copyright	&copy;	&#169;
ª	feminine ordinal indicator	&ordf;	&#170;
«	angle quotation mark (left)	&laquo;	&#171;
¬	negation	&not;	&#172;
	soft hyphen	&shy;	&#173;
®	registered trademark	&reg;	&#174;

™	trademark	&trade;	&#8482;
¯	spacing macron	&macr;	&#175;
°	degree	&deg;	&#176;
±	plus-or-minus	&plusmn;	&#177;
²	superscript 2	&sup2;	&#178;
³	superscript 3	&sup3;	&#179;
´	spacing acute	&acute;	&#180;
µ	micro	&micro;	&#181;
¶	paragraph	&para;	&#182;
·	middle dot	&middot;	&#183;
¸	spacing cedilla	&cedil;	&#184;
¹	superscript 1	&sup1;	&#185;
º	masculine ordinal indicator	&ordm;	&#186;
»	angle quotation mark (right)	&raquo;	&#187;
¼	fraction 1/4	&frac14;	&#188;
½	fraction 1/2	&frac12;	&#189;
¾	fraction 3/4	&frac34;	&#190;
¿	inverted question mark	&iquest;	&#191;
×	multiplication	&times;	&#215;
÷	division	&divide;	&#247;

HTML TAGS LIST

Tag	Description
<!--...-->	Defines a comment
<!DOCTYPE>	Defines the document type
<a>	Defines a hyperlink

HTML5&CSS3

<abbr>	Defines an abbreviation
<acronym>	Not supported in HTML5
<address>	Defines contact information for the author/owner of a document/article
<applet>	Not supported in HTML5
<area>	Defines an area inside an image-map
<article>New	Defines an article
<aside>New	Defines content aside from the page content
<audio>New	Defines sound content
<b>	Defines bold text
<base>	Specifies the base URL/target for all relative URLs in a document
<basefont>	Not supported in HTML5
<bdi>New	Isolates a part of text that might be formatted in a different direction from other text outside it
<bdo>	Overrides the current text direction
<big>	Not supported in HTML5
<blockquote>	Defines a section that is quoted from another source
<body>	Defines the document's body
<br>	Defines a single line break
<button>	Defines a clickable button
<canvas>New	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<caption>	Defines a table caption
<center>	Not supported in HTML5
<cite>	Defines the title of a work
<code>	Defines a piece of computer code
<col>	Specifies column properties for each column within a <colgroup> element
<colgroup>	Specifies a group of one or more columns in a table for formatting
<command>New	Defines a command button that a user can invoke
<datalist>New	Specifies a list of pre-defined options for input controls
<dd>	Defines a description of an item in a definition list
<del>	Defines a text that has been deleted from a document
<details>New	Defines additional details that the user can view or hide
<dfn>	Defines a definition term
<dir>	Not supported in HTML5
<div>	Defines a section in a document
<dl>	Defines a definition list
<dt>	Defines a term (an item) in a definition list
<em>	Defines emphasized text
<embed>New	Defines a container for an external (non-HTML) application.

HTML5&CSS3

<fieldset>	Groups related elements in a form
<figcaption>New	Defines a caption for a <figure> element
<figure>New	Specifies self-contained content
<font>	Not supported in HTML5
<footer>New	Defines a footer for a document or section
<form>	Defines an HTML form for user input
<frame>	Not supported in HTML5
<frameset>	Not supported in HTML5
<h1> to <h6>	Defines HTML headings
<head>	Defines information about the document
<header>New	Defines a header for a document or section
<hgroup>New	Groups heading (<h1> to <h6>) elements
<hr>	Defines a thematic change in the content
<html>	Defines the root of an HTML document
<i>	Defines a part of text in an alternate voice or mood
<iframe>	Defines an inline frame
<img>	Defines an image
<input>	Defines an input control
<ins>	Defines a text that has been inserted into a document
<keygen>New	Defines a key-pair generator field (for forms)
<kbd>	Defines keyboard input
<label>	Defines a label for an input element
<legend>	Defines a caption for a <fieldset>, <figure>, or <details> element
<li>	Defines a list item
<link>	Defines the relationship between a document and an external resource
<map>	Defines a client-side image-map
<mark>New	Defines marked/highlighted text
<menu>	Defines a list/menu of commands
<meta>	Defines metadata about an HTML document
<meter>New	Defines a scalar measurement within a known range (a gauge)
<nav>New	Defines navigation links
<noframes>	Not supported in HTML5
<noscript>	Defines an alternate content for users that do not support client-side scripts
<object>	Defines an embedded object
<ol>	Defines an ordered list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<output>New	Defines the result of a calculation
<p>	Defines a paragraph

<param>	Defines a parameter for an object
<pre>	Defines preformatted text
<progress>New	Represents the progress of a task
<q>	Defines a short quotation
<rp>New	Defines what to show in browsers that do not support ruby annotations
<rt>New	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>New	Defines a ruby annotation (for East Asian typography)
<s>	Defines text that is no longer correct
<samp>	Defines sample output from a computer program
<script>	Defines a client-side script
<section>New	Defines a section in a document
<select>	Defines a drop-down list
<small>	Defines smaller text
<source>New	Defines multiple media resources for media elements (<video> and <audio>)
<span>	Defines a section in a document
<strike>	Not supported in HTML5
<strong>	Defines important text
<style>	Defines style information for a document
<sub>	Defines subscripted text
<summary>New	Defines a visible heading for a <details> element
<sup>	Defines superscripted text
<table>	Defines a table
<tbody>	Groups the body content in a table
<td>	Defines a cell in a table
<textarea>	Defines a multiline input control (text area)
<tfoot>	Groups the footer content in a table
<th>	Defines a header cell in a table
<thead>	Groups the header content in a table
<time>New	Defines a date/time
<title>	Defines a title for the document
<tr>	Defines a row in a table
<track>New	Defines text tracks for media elements (<video> and <audio>)
<tt>	Not supported in HTML5
<u>	Defines text that should be stylistically different from normal text
<ul>	Defines an unordered list
<var>	Defines a variable
<video>New	Defines a video or movie
<wbr>New	Defines a possible line-break

HTML5&CSS3

HTML 5 HISTORY:
HTML5 is a markup language, has been come into existence around January 2008. The two measure organization is involving in developing of HTML5 since its initiating time. One is W3C (World Wide Web Consortium) and another one is WHATWG (Web Hypertext Application Technology Working Group). According to these organizations, they have been working on the HTML5 since initial time. So HTML5 language is still under development. There is more about to come yet in HTML5.

During the development of HTML5, It was announced that the HTML5 will reach the W3C recommendation till at the end of 2010. But the last call didn't match till the target date. Now according to W3C the HTML5 will reach its full recommendation last by 2014.

Where according to WHATWG the last call for HTML5 Specification was in October 2009. Then suddenly the amazing changes in decision the WHATWG started to work on versioned development of HTML, and with abounding its HTML5 Project. Later in January 2011, it renamed the HTML5 Stander to HTML5.

On 18 January 2011, the W3C introduce a logo to represent the HTML5 interest. While presenting its logo to publicly, W3C announced that, the logo can be used for general purpose.

WHAT IS HTML 5?
HTML5 is the HTML5est version of Hyper Text Markup Language. The first web browser introduced in 1993 and name was MOSAIC. The development of MOSAIC was at the NCSA (National Center of Supercomputing Applications). Later it was discontinued to development on 7th of January 1997. Still the people were using the nonstandard version of HTML.

The standard version came into existence in 1995, when HTML 2.0 was an-announced. Later after two years HTML 3.0 and after two years HTML 4.01 was announced. And still we are using the milestone of HTML 4.01. The first Draft of HTML5 Was announced in January 2008. And amazingly HTML5 has a broad browser support. Though the HTML5 is still under developing phase. And a lot of organizations is working and planning on the development of HTML5.

We can't expect the HTML5 may be the future of Web Designing, but we can say that this is the present of Web designing. Before development of HTML5, we were in compulsion to work on Photoshop and Flash application, but with the development of HTML5, these affords has been reduced. Much more long script code can be done with a simple tagging. As we can use <details> and <summary> tag for show and hide function of java Script. We need not to put a long affords to code this thing. Apart from this features we can use the 3D image with <canvas>, the special designed paragraph with <article> and many more.

HTML5&CSS3

HTML or Hypertext Markup Language is the most widely used language on Web. Technically, HTML is not a programming language, but rather a markup language. This tutorial gives a complete understanding on HTML.

HTML5 feature groups:

HTML5 is making the web platform more powerful in a number of different areas.

1. Client-Side storage (Expect the unexpected)
Client-Side storage, web storage, offline Storage, Local Storage. Local Storage is intended to be used for storing and retrieving data in html pages from the same domain. The data can be retrieved from all the windows in the same domain even if the browser is restarted

2. Real Time Communications :(Stay connected)
Web Workers:
A web worker is a JavaScript running in the background, without affecting the performance of the page.

Web Socket
HTML5 Web Socket defines a bi-directional, full-duplex communication channel that operates through a single TCP socket over the Web

3. File / Hardware Access (Deeper integration with the Operating System)
Native Drag & Drop
Drag and Drop (DnD) is powerful User Interface concept which makes it easy to copy, reorder and deletion of items with the help of mouse clicks.

File System APIs
Reading and manipulating files: File/Blob, FileList, File Reader

Geolocation:
HTML5 Geolocation is used to locate a user's position.

Device Orientation:
Device orientation is a JavaScript API which makes it possible for apps to utilize data from accelerometers, gyroscopes or compasses in the smartphone.

Speech Input:
You can add speech input to any input element by simply using the x-webkit-speech attribute. <input type="text" x-webkit-speech />

4. Semantics & Markup (More meaningful elements)

HTML5&CSS3

Better semantic tags like <article> <aside> <details> <footer> <header>etc..!

Form field types on mobile

- ✓ android keyboard on input type text Android Device type="number"
- ✓ android keyboard on input type number Android Device type="tel"
- ✓ iPhone keyboard on input type email iPhone Device type="email"

5. HTML5-MULTIMEDIA (playing video and audio is easier than ever)
HTML5 <video>

- ✓ HTML5 provides a standard for playing video files
- ✓ HTML5 <audio>
- ✓ HTML5 provides a standard for playing audio files

6. HTML5-GRAPHICS (2D and 3D Effects)
Canvas 2D

- ✓ <canvas> has a wealth of features, like:
- ✓ drawing shapes,
- ✓ filling colors,
- ✓ creating gradients and patterns,
- ✓ rendering text,
- ✓ copying images, video frames, and other canvases,

Canvas 3D (WebGL)
WebGL library to explore 3d geometries, lighting, materials, textures, grouping, scenes, imported models, and events. WebGL (Web Graphics Library)

7. PERFORMACE AND INTEGRATION (Improvements to the core platform)

- ✓ Use web storage in place of cookies
- ✓ Use CSS transitions instead of JavaScript animation
- ✓ Use client-side databases instead of server round-trips

HTML5 document structure:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="utf-8" />
<title>Page Title</title>
</head>
<body>
  <header>...</header>
  <nav>...</nav>
  <section>
    <header>...</header>
    <article>
      <p>...</p>
    </article>
     <footer>...</footer>
  </section>
  <aside>...</aside>
  <footer>...</footer>
</body>
</html>
```

<! DOCTYPE HTML>

In previous chapter you leant about the comment tag, which is basically used in every program. Now we will learn in this chapter about DOCTYPE element. In HTML previous version there was need of a DTD declaration with DOCTYPE assuming like (<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transi- tional.dtd">), which was a little bit

Complex to remember even and slightly mistakes may have occur during writing such a long DOCTYPE element. So with the development of HTML5 this feature of HTML has been enhanced.

<!DOCTYPE html>

DOCTYPE is such an element, which tells the browser about the html version, DOCTYPE is not a HTML tag, but for the Proper SEO purpose and introducing the version of HTML to the browser the tag is very useful. HTML5 is not based on SGML, so there is no need to declaration of DTD with the DOCTYPE tag. DOCTYPE tag is defined at the top of the page.

Syntax for DOCTYPE

```
<!DOCTYPE html
```

Example

```
<!DOCTYPE html>
<html>
<head>
<title>
Page title will go here
</title>
</head>
<body>
This is test page
</body>
</html>
```

HTML5&CSS3

HTML5 ELEMENTS

HTML5 has been updated with few HTML5 ELEMENTS, which makes HTML5 more efficient markup language. We can reduce our external affords to use these HTML5 ELEMENTS. The Internet has changed since last development of HTML 4.01, so there are certain changes has been made in browser to use it more correctly. Few OLD ELEMENTS has been removed, where few HTML5 ELEMENTS has been updated in HTML5.

<aside>	Defines content aside from the page content
<audio>	Defines sound content
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<canvas>	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<command>	Defines a command button that a user can invoke
<datalist>	Specifies a list of pre-defined options for input controls
<defines>	article an article
<details>	Defines additional details that the user can view or hide
<embed>	Defines a container for an external application or interactive content (a plug-in)
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<figcaption>	Defines a caption for a <figure> element
<footer>	Defines a footer for a document or section
<header>	Defines a header for a document or section
<hgroup>	Groups a set of <h1> to <h6> elements when a heading has multiple levels
<keygen>	Defines a key-pair generator field (for forms)
<mark>	Defines marked/highlighted text

<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links
<output>	Defines the result of a calculation
<progress>	Represents the progress of a task
<ruby>	Defines a ruby annotation (for East Asian typography)
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<rp>	Defines what to show in browsers that do not support ruby annotations
<section>	Defines a section in a document
<source>	Defines multiple media resources for <video> and <audio>
<summary>	Defines a visible heading for a <details> element
<track>	Defines text tracks for <video> and <audio>
<time>	Defines a date/time
<video>	Defines a video or movie

<wbr>	Defines a possible line-break

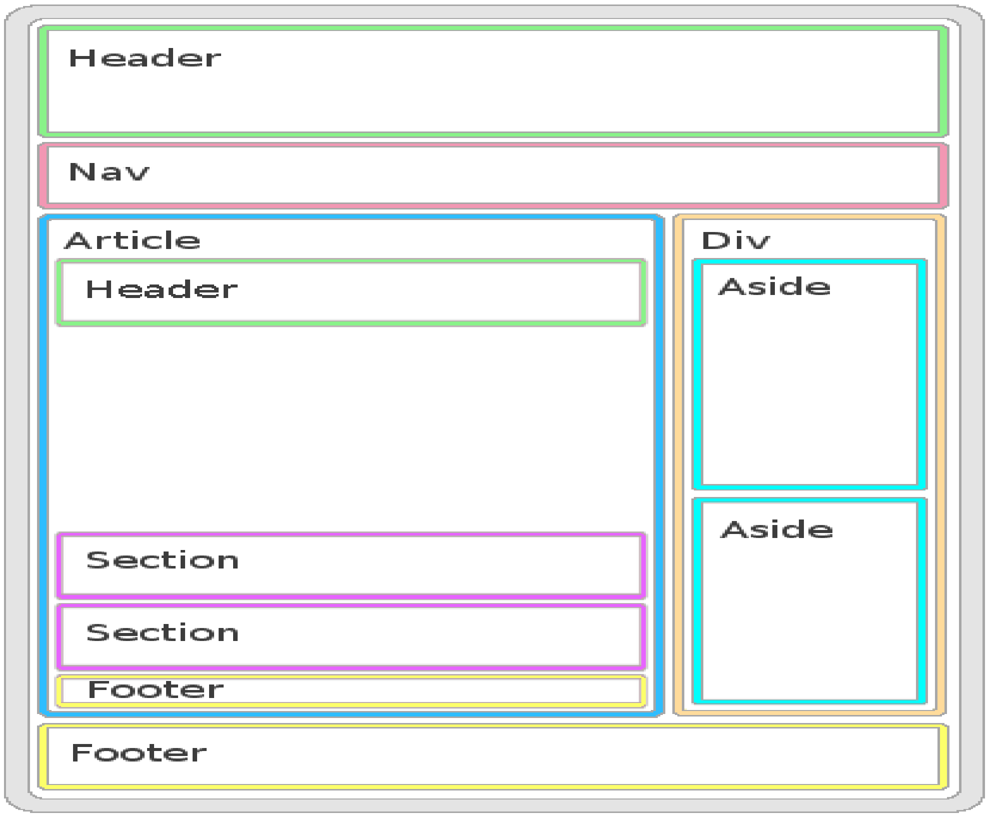
HTML5 Semantic/Structural Elements
HTML5 offers HTML5 elements for better structure:

1. <section>
2. <article>
3. <header>
4. <footer>
5. <hgroup>
6. <aside>
7. <command>
8. <details>
9. <summary>
10. <figure>
11. <figcaption>
12. <nav>
13. <wbr>
14. <bdi>
15. <bdo>
16. <ruby>
17. <rt>
18. <rp>

Tag	**Description**
<section>	Defines a section in a document
<article>	Defines an article
<header>	Defines a header for a document or section
<footer>	Defines a footer for a document or section
<hgroup>	Groups a set of <h1> to <h6> elements
<aside>	Defines content of related surrounding content
<command>	Defines a command button
<details>	Def. additional details that the user can view/hide
<summary>	Defines a visible heading for a <details> element
<figure>	Spe. Self-contained content, like photos, code, listings, etc.
<figcaption>	Defines a caption for a <figure> element
<nav>	Defines navigation links
<wbr>	Defines a possible line-break
<bdi>	Isolates a part of text formatted in a different direction
<bdo>	It is used to override the current text direction
<ruby>	Defines a ruby annotation(for East Asian typography)
<rt>	Defines an explanation/pronunciation of characters

<rp>	Defines what to show in browsers

STRUCTURE OF HTML5 SEMANTICS



HTML5 Document
The following tags have been introduced for better structure:

Section: This tag represents a generic document or application section. It can be used together with h1-h6 to indicate the document structure.

Article: This tag represents an independent piece of content of a document, such as a blog entry or newspaper article.

Aside: This tag represents a piece of content that is only slightly related to the rest of the page.

Header: This tag represents the header of a section.

Footer: This tag represents footer for a section and can contain information about the author, copyright information, et cetera.

Nav: This tag represents a section of the document intended for navigation.

Dialog: This tag can be used to mark up a conversation.

Figure: This tag can be used to associate a caption together with some embedded content, such as a graphic or video.
Etc………….

<section>
The <section> tag defines sections in a document. Such as chapters, headers, footers, or any other sections of the document. It is a paired tag.

Syntax:
<section>.........................</section>

Attributes:
Element-Specific Attributes None.

Example:
```
<!DOCTYPE html>
<html>
<body>
<section>
<article>
<header>CSS</header>
A style sheet consists of a list of rules. Each rule or rule-set consists of one or more selectors, and a declaration block. A declaration-block consists of a list of declarations in braces. Each declaration itself consists of a property, a colon (:), and a value. If there are multiple declarations in a block, a semi-colon (;) must be inserted to separate each declaration.
<footer> &copy; &reg; reserved W3C. </footer>
 </article>
</section>
<section>
 <h1>CSS3</h1>
<p>The development of CSS3 is going to be split up into 'modules'. Some of these modules are: The Box Model Lists Module Hyperlink Presentation Speech Module Backgrounds and Borders Text Effects Multi-Column Layout.</p>
</section>
</body>
</html>
```

<header>

The HEADER Element is commonly used for defining the header for a particular document or section. The HEADER Element also can be used for introductory content of a container and may be also used for navigational links. You may have multiple HEADER Elements in a single document. The HEADER Element can’t be used within the footer, address and another header element.

Syntax:
<header>-----------------</header>

Attributes:
Element-Specific Attributes are none.

Example:
```
<!DOCTYPE html>
<html>
<body>
<article>
  <header>
  <h1>Search Engine Optimization</h1>
  </header>
  <p>As an Internet marketing strategy, SEO considers how search engines work, what people search for, the actual search terms or keywords typed into search engines and which search engines are preferred by their targeted audience</p>
</article>
</body>
</html>
```

<footer>
FOOTER Element is used define the FOOTER area of the page. A FOOTER Element can contain the copyright information, terms and uses, author’s information, developers and etc. A FOOTER Element may content many of other elements also. It is a paired tag.

Syntax: <footer>-----------------</footer>

Attributes:
Element-Specific Attributes are None.

Example:
```
<!DOCTYPE html>
<html>
<body>
<article>
```

```
<p>Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. </p>
<p>Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.</p>
</article>
<footer>
 &copy 2012 RIA Internet Apps &reg
</footer>
</body>
</html>
```

HTML5 <article> Tag:
An ARTICLE Tag is such an element of HTML5, which can be used to write the article. If a user wants to write a part of the website in different style and looking in different manner, the ARTICLE Tag can be used there to rep- resent it in a different way. The ARTICLE Tag is used to define the in dependent content in a note. The ARTICLE Tag is very useful when you write an article, a blog or a forum post and etc. ARTICLE Tag of HTML5 can contain any of element which is require to create the proper content, either it may be the Para tag, span tag or it may be header, footer section. It is paired tag.

Syntax: <article>------------------</article>

Attributes:
Element-Specific Attributes None

Example

```
<article>
  <h4>A really awesome article</h4>
  <p>Lots of awesome text. It is good Article</p>
</article>
```

Example

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>HTML5 article example</title>
</head>
<body>
<article>
<h1>HTML5 article element</h1>
<p>HTML5 article element represents independent item like a blog entry in an web document.</p>
```

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

Example

```
<!Doctype HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>Example of article tag.</title>
</head>
<body>
<p><b>Example of article <article> tag in HTML5.</b></p>
<article>
<p>
It is global services company that understands businesses and aims to deliver value to its customers through its software solutions and services. </p>
</article>
</body>
</html>
```

<hgroup>

The hgroup element is typically used to group a set of one or more h1-h6 elements -to group,The HTML <hgroup> tag is used for defining the header of an HTML document or section. such as subheadings, alternative titles, or taglines. It is a paired tag.

Syntax: <hgroup> ---------</hgroup>

Attributes:
Element-Specific Attributes are None.

Example:

```
<!DOCTYPE html>
<html>
<body>
<hgroup>
<h1>Welcome to my NareshTech</h1>
<h2>All Web Technologies Train Here</h2>
<h3> like HTML CSS JS jQuery </h3>
</hgroup>
<p>The rest of the content...</p>
</body>
</html>
```

<aside>
An ASIDE Element is used to define the text surrounded by its familiar con- tent. ASIDE Element has its individual importance to creating some index related to the topic. Mainly ASIDE Tag is used in sidebar of a page, when- ever user clicks on the index, it redirects the user on the related content. ASIDE Element also can be used to represent some block quote content in an article. It is a paired tag..!!

Syntax: <aside>-------------</aside>

Attributes:
Element-Specific Attributes --> None

Example:
```
<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset=utf-8>
</head>
<body>
<p>HTML5 is Web Environment or platform for future development</p>
<aside>
<h4>Cascading Style Sheets</h4>
<p>It is another advanced look and feel for designing.</p>
</aside>
</body>
</html>
```

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>
Good Example for ASIDE
</title>
</head>
<body>
<aside style="font-size:larger;font-style:italic;color:blue;float:right;width:120px;">
HTML5 is Web platform. It is collection of technologies. It is from WHATWG and W3C.
</aside>
```

<p>The aside element is HTML5 to HTML5 and it can be used in two different contexts. Basically, the context of the aside element is based on whether or not it inside or outside the article element.</p>
<p>The HTML article tag is used to represent an article. More specifically, the content within the article tag is independent from the other content on the site. This could be a forum post, a magazine or HTML5spaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content..</p>
</body>
</html>

<command>
COMMAND Element is basically used for commanding code of HTML5. As if we want to save a file, open a file or anything else that shows the command function will be prompted with COMMAND Element. It can be used for radio buttons, checkboxes and command button to invoke the particular function. COMMAND Element is not supported by any of the browsers yet. This Element is still under development. The COMMAND Element can be used inside the MENU Element or outside anywhere in the BODY Element. It is a paired tag.

Syntax:
<command>------------------</command>

Attributes:

Attribute	Value	Description
checked	checked	Specifies that the command should be checked when the page
disabled	disabled	Specifies that the command should be disabled.
icon	URL	Specifies an image that represents the command.
label	Text	Specifies the name of the command, as shown to the user.

Example:
```
<menu>
<command onclick="alert('Hello World')">
Click Me!</command>
</menu>
```

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>
Good Example for Command
</title>
<script>
```

```
function save()
{
alert("Command...Success....");
}
</script>
</head>
<body>
<menu>
<input type="command"
label="Save" onclick="save()">Save</command>
</menu>
</body>
```

<details>

DETAILS Element is used to invoke the show and hide function of HTML5. Before the development of HTML5 we were using the JavaScript for show and hide function. But now it is too easy to use this function with this DETAILS and SUMMARY Tag of HTML5. It is a paired tag.

Syntax: <details>------------------</details>

Attributes

Attribute	Value	Description
open	open	Specifies that the details should be visible (open) to the user

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
</head>
<body>
<details>
<summary>Copyright 1999-2012.</summary>
<p> - NareshiTech. All Rights Reserved.</p>
<p>All contents are related to IT resources and Training and Development</p>
</details>
</body>
</html>
```

Example

```
<!DOCTYPE HTML>
<html lang="en-US">
```

```
<head>
<meta charset=utf-8>
</head>
<body>
<details open="open">
<summary>Copyright 1999-2015.</summary>
<p> - NareshiTech. All Rights Reserved.</p>
<p>All contents are related to IT resources and Training and Development</p>
</details>
</body>
</html>
```

<summary>
The <summary> tag defines a visible heading for the <details> element. The heading can be clicked to view/hide the details. It is a paired tag.

Syntax: <summary>------------------</summary>

Note: The <summary> tag is currently only supported in Chrome.
Note: The <summary> element should be the first child element of the <details> element.

Attributes
Element-Specific Attributes are None.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
</head>
<body>
<details>
<summary>Nareshit Description</summary>
<p>Naresh i Technologies</p>
<p>Leader in IT Training</p>
<p>Ameerpet-vist-www.nareshit.com</p>
<p>Hyderabad</p>
</details>
</body>
</html>
```

<figure>

FIGURE Element is used to call an image file inside the fixed container. FIGURE Element has its fix container properties which is relevant to the remaining text. Whenever a programmer calls an external image file inside the FIGURE Element, It automatically adjusts the paragraph text and figure alignment. It is a paired tag.

Note: It is supported in Internet Explorer 9, Firefox, Opera, Chrome, and Safari.
Note: Internet Explorer 8 and earlier versions do not support the <figure> tag.

Syntax: <figure>---------------------</figure>

Attributes: Element-Specific Attributes are none.

Example:
```
<!DOCTYPE html>
<html>
<body>
<p>HTML5 will be the HTML5 standard for HTML, XHTML, and the HTML DOM.
The previous version of HTML came in 1999. The web has changed a lot since then.
HTML5 is still a work in progress. However, most modern browsers have some HTML5 support...</p>
<figure>
  <img src="html5.png" title="WebPlatform" width="300" height="250" />
</figure>
</body>
</html>
```

<figcaption>
FIGCAPTION Element is used to put additional information about the image. As if we have described about the image, we don’t need to define an paragraph to define the description. It may slightly complex also to use another paragraph and style it with the displayed image. So with FIG- CAPTION Element, we can exactly put a description and it will automatically adjust its surrounding style according the image. It is a paired tag.

Syntax: <figcaption>----------------</figcaption>

Attributes:
Element-Specific Attributes are None.

Example:
```
<!DOCTYPE html>
<html>
<body>
<p>HTML5 will be the HTML5 standard for HTML, XHTML, and the HTML DOM.
```

```
The previous version of HTML came in 1999. The web has changed a lot since then.
HTML5 is still a work in progress. However, most modern browsers have some HTML5 support...</p>
<figure>
  <img src="html5.png" title="Web Platform" width="300" height="250" />
<figcaption> It is HTML5 Logo from W3C and WHATWG</figcaption>
</figure>
</body>
</html>
```

<nav>
The HTML <nav> tag is used for declaring a navigational section of the HTML document. Websites typically have sections dedicated to navigational links - links that enable the user to navigate the site. These links should be placed inside a <nav> tag. It is a paired tag.

Syntax: <nav>--------------------</nav>

Attributes:
Element-Specific Attributes None.

Example
```
<!DOCTYPE html>
<html>
<body>
<nav>
<a href="http://www.nareshit.com">NareshIT</a> |
<a href="http://www.nareshit.in">NaershIN</a> |
<a href="http://www.seshajobs.com">NiTJobs</a> |
<a href="http://www.nacreservices.com">TalentTest</a>
</nav>
</body>
</html>
```

<wbr>
The WBR Element is used to change the default behavior of browser of line breaking. If the sentence is too long than the container, will be beaked for proper displaying. If you don't want it to be happened like that, can use the WBR Element. This can be more useful for email id writing, and another long sentence. It is non-paired Tag

Note: The <wbr> tag is supported in all major browsers, except Internet Explorer.

Syntax: <wbr>

Attributes:
Element-Specific Attributes None.

Example:
```
<!DOCTYPE html>
<html>
<head>
<title>Title name will go here</title>
</head>
<body>
<p>This is my email id : <wbr>nithtml5css3@gmail.com<wbr></p>
</body>
</html>
```

Example:
```
<!DOCTYPE html>
<html>
<head>
<title>
wbr tag
</title>
</head>
<body>
<p>http://this<wbr>.is<wbr>.a<wbr>.really<wbr>.long<wbr>.example<wbr>.com/With<wbr
>/deeper<wbr>/level<wbr>/pages<wbr>/deeper<wbr></p>
</body>
</html>
```

HTML <bdo> Tag
bdo stands for Bi-Directional Override. The <bdo> tag is used to override the current text direction. This can be useful when displaying Hebrew, Arabic, and other languages/scripts that are written from right to left. It is paired Tag.

Syntax: <bdo>......................</bdo>

Attributes

Attribute	**Value**	**Description**
dir	ltr	Specifies the text direction
	rtl	Specifies the text direction

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
```

```
<head>
<meta charset=utf-8>
</head>
<body>
<bdo dir="ltr">
How to override text direction?
I think you already know!
</bdo>
<br/>
<bdo dir="rtl">
How to override text direction?
I think you already know!
</bdo>
 </body>
</html>
```

<bdi>

The BDI Element stand for Bi-Directional Isolation Element, which is one of the best feature of HTML5, especially when someone wants to display a text in the Bi-direction way around the remaining text. Whenever we write the aroma, Hebrew or Urdu fonts, it shows from the opposite side of the general fonts. So browser behavior directly changes for that particular text. To use of BDI is exactly has been implemented for better performance of those texts, that's can be read easily by the user. The more confusion can be cleared about the BDI Element in the example. Try our try it editor to have better understanding.

Syntax:

<bdi>..............................</bdi>

Attributes

Element-Specific Attributes are None.

Example:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<title>
bdi tag
</title>
<meta charset=utf-8>
</head>
<body>
<p dir="ltr">This arabic word <bdi>ARABIC_PLACEHOLDER</bdi> is automatically displayed
right-to-left. </p>
```

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

<ruby>

The RUBY Element is used to properly rendering the East Asian's languages. According the W3C specification it has been described as "The RUBY Element allows spans of phrasing content to be marked with ruby annotations". The RUBY Element is basically used to give a pronunciation help in a phonetic script for Chinese, Japanese and Korean languages. The RUBY Element is used to display the text on the head of the base text, and auto renders the base text and RUBY text in well manner to enhance the user experience in reading the document. There are mainly three attributes can be called with RUBY Element. rt (ruby text), rp (ruby parenthesis) and rb (ruby base).

Syntax:
<ruby>--------------------</ruby>

Attributes:
Element-Specific Attributes None.

Example:

```
<!DOCTYPE html>
<html lang="en-US">
<body>
 <p>...<ruby>漢<rt>かん</rt>字<rt>じ</rt></ruby>...</p>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang="en-IN">
<head>
<title>
bdi tag
</title>
<meta charset="utf-8">
</head>
<body>
<ruby>
  漢 <rp>(</rp><rt>Kan</rt><rp>)</rp>
  字 <rp>(</rp><rt>ji</rt><rp>)</rp>
</ruby>
</body>
```

</html>

<rt> (Ruby Text)
It marks the Ruby Text component of a ruby annotation.Ruby annotations are used in East Asian typography. It is a paired tag.

Syntax:
<rt>..................</rt>

Attributes:
Element-Specific Attributes None.

Example:
```
<!DOCTYPE html>
<html lang="en-US">
<body>
<ruby>
 <rt> </rt>
</ruby>
</body>
</html>
```

<rp> (ruby parenthesis)
The HTML <rp> is used in ruby annotations for the benefit of browsers that don't support ruby annotations. It is a paired tag.

Syntax:
<rp>---------------------</rp>

Attributes:
Element-Specific Attributes None.

Example:
```
<!DOCTYPE html>
<html lang="en-US">
<head>
<title>
rp tag
</title>
<meta charset="utf-8">
</head>
<body style="font: 75% Lucida Grande, Trebuchet MS">
  きのうの豪雨で山の水源地は<ruby>氾濫<rp>（</rp>
```

```
   <rt>はんらん</rt><rp>）</rp></ruby>し、濁流
   <ruby>滔々<rp>（</rp><rt>とうとう</rt><rp>）</rp>
   </ruby>と下流に 集り、猛勢一挙に橋を破壊し、どうどうと
   響きをあげる激流が、<ruby>木葉微塵<rp>（</rp>
   <rt>こっぱみじん</rt><rp>）</rp></ruby>に<ruby>橋桁
   <rp>（</rp><rt>はしげた</rt><rp>）</rp></ruby>
   を跳ね飛ばしていた。
  </body>
</html>
```

<rtc>

The HTML <rtc> Element embraces semantic annotations of characters presented in a ruby of <rb> elements used inside of <ruby> element. <rb> elements can have both pronunciation (<rt> and semantic (<rtc>) annotations. It is a paired tag.

Syntax:

<rtc>.....................................</rtc>

Example:

```
<!DOCTYPE html>
<html lang="en-IN">
<head>
<title>
rtc tag
</title>
<meta charset="utf-8">
</head>
<body>
<ruby>
  <rb>旧</rb>
  <rb>金</rb>
  <rb>山</rb>
  <rt>jiù</rt>
  <rt>jīn</rt>
  <rt>shān</rt>
  <rtc>San Francisco</rtc>
</ruby>
</body>
</html>
```

HTML <rb> Tag

The HTML <rb> marks the base text component of a ruby annotation. Ruby annotations are used in East Asian typography. The basic tag is written like this <rb></rb> with the base text written between the opening and closing tags. The tags must be nested inside a <ruby> element. The closing tag may be omitted if the <rb> element is immediately followed by an <rb>, <rt>, <rtc> or <rp> element, or if there is no more content in the parent element. It is a non-paired tag.

Syntax:
<rb> ....</rb>

Example:

```
<!DOCTYPE html>
<html lang="en-IN">
<head>
<title>
rb tag
</title>
<meta charset="utf-8">
</head>
<body>
<ruby>
 <rb>旧
 <rb>金
 <rb>山
 <rt>jiù
 <rt>jīn
 <rt>shān
 <rtc>San Francisco
</ruby>
</body>
</html>
```

HTML5 NEW INLINE ELEMENTS

HTML 5 introduces new elements to help indicate basic elements such as times or numbers.

<mark>

This denotes that a bit of text is marked in some way. You could, for example, use this to mark search terms in a list of results.

<meter>

This can be used to indicate a figure of some sort. It can have multiple attributes including: value, min, max, low, high, and optimum.

<progress>

This can be used to show a progress bar of some sort. It has a couple of attributes: value and max. The max attribute can be omitted.

<time> -
You can use this to represent time or date in your block of text.

<mark> Tag
The MARK Element is basically used for showing a marked text background. The MARK Element typically highlights the text with another background color, which attracts the reader and focus the particular text for the reference using. In the present scenario many of search engines are using these features of HTML5. When they search the particular text, they put that text into the MARK Element and then represent it on the screen. The MARK Element can be used for enhance the reading of user experience. It is a paired tag.

Syntax:
<mark>..............................</mark>

Attributes:
Element-Specific Attributes are none.

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<body>
<p>Do not forget to learn about WEB <mark>HTML5</mark> Today </p>
</body>
</html>
```

Example:
```
<!DOCTYPE html>
<html>
<head>
<title>Title name will go here</title>
</head>
<body>
<p>You are using the <mark>Mark</mark> element</p>
</body>
</html>
```

Example:(CSS)
```
<head>
<style type='text/css'>
```

```
mark
{
background-color:#FF0099;
}
</style>
</head>
<body>
<mark> Naresh i Technologies</mark>
</body>
```

<meter> Element:
The METER Element is basically used for scalar measurement for known range or known a fractional value. This can be used for disk usage, relevance query status results and etc. The METER Element can't be used if we don't have the known range. There are six attributes allowed in METER Element: value, min, max, high, low and optimum. It is a paired tag.

Syntax:
<meter>..............................</meter>

Note: The <meter> tag should not be used to indicate progress.

Attributes:

Attribute	**Value**	**Description**
form	form_id	Specifies one or more forms
high	number	Specifies the range that is considered to be a high
low	number	Specifies the range that is considered to be low
max	number	Specifies the maximum value of the range
min	number	Specifies the minimum value of the range
optimum	number	Specifies value is the optimal value for the gauge
value	number	Specifies the current value of the gauge

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<p> The following meter full 20%..!!</p>
<meter value="2" min="0" max="10">
<p style='color:red'>OOPs Your Browser not supporting meter element or tag...</p>
</meter>
<br/>
```

```
<p>The following meter full 40%..!!</p>
<meter value="4" min="0" max="10">
<p style='color:red'>OOPs Your Browser not supporting meter element or tag...</p>
</meter>
<br/>
<p style='color:red'>The following meter full 80%..!!</p>
<meter value="8" min="0" max="10">
<p style='color:red'>OOPs Your Browser not supporting meter element or tag...</p>
</meter>
<br/>
</body>
</html>
```

Note:
1. If value is higher than high, the gauge is Yellow: (When Low available)
2. When value is lower than low, if optimum is lower than low, the gauge is Green
3. If Value is more than high then optimum is Red (When Max Available)

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<meter low="69" high="80" max="100" value="84">
<p style='color:red'>OOPs Your Browser not supporting meter element or tag...</p>
</meter>
<meter high="80" max="100" value="84">
<p style='color:red'>OOPs Your Browser not supporting meter element or tag...</p>
</meter>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<meter value=3 min=0 max=4 low=1 high=3>3  of 4>
<p style='color:red'>OOPs Your Browser not supporting meter element or tag...</p>
```

```
</meter><br/>
<meter value=4 min=0 max=4 low=1 high=3>
<p style='color:red'>OOPs Your Browser not supporting meter element or tag...</p>
</meter><br/>
<meter value=61 min=0 max=100  low=73 high=87 optimum=100>
<p style='color:red'>OOPs Your Browser not supporting meter element or tag...</p>
</meter>
</body>
</html>
```

HTML5 <progress> Tag

The HTML <progress> tag is used for representing the progress of a task. This element could be used in conjunction with JavaScript to display the progress of a task or process as it is underway. The <progress> is not suitable for representing a gauge (such as disk space usage or a tally of votes). To represent a gauge, use the <meter> tag instead.

Syntax: <progress>.................</progress>

Attributes:

Attribute	Value	Description
max	number	Specifies how much work the task requires in total
value	number	Specifies how much of the task has been completed

Example:

```
<!DOCTYPE html>
<html>
<body>
Downloading progress:
<progress value="22" max="100">
<p>OOPs Your Browser not supporting… </p>
</progress>
</body>
</html>
```

Example:

```
<html>
<head>
<title>
Set Intervals
</title>
<script>
setInterval("fun1()",1000);
function fun1()
```

```
{
var d=new Date
str=d.getHours()+":"+d.getMinutes()+":"+d.getSeconds()
document.getElementById('sp1').innerHTML=str
}
</script>
</head>
<body>
<span id="sp1" style="color:red;font-size:30">
</span>
</body>
</htm>
```

Example:

```
<body>
<script type='text/javascript'>
window.onload = function()
{
var bar = document.getElementById("bar"),
fallback = document.getElementById("fallback"),
loaded = 0;
var load = function()
{
loaded += 10;
bar.value = loaded;
if(loaded == 100)
{
clearInterval(dummyLoad);
}
};
var dummyLoad = setInterval(function()
{
load();
} ,100);
}
</script>
<p>
<progress id="bar" value="0" max="100">
<span id="fallback"></span> </progress></p>
</body>
```

Example:

```
 <body>
```

```
<progress id="progressBar" value="0" max="100" style="width:300px;"></progress>
<span id="status"></span>
<h1 id="finalMessage"></h1>
<script type="text/javascript" language="javascript">
function progressBarSim(al)
{
var bar = document.getElementById('progressBar');
var status = document.getElementById('status');
status.innerHTML = al+"%";
bar.value = al;
al++;
var sim = setTimeout("progressBarSim("+al+")",10);
if(al == 100)
{
status.innerHTML = "100%";
bar.value = 100;
clearTimeout(sim);
var finalMessage = document.getElementById('finalMessage');
finalMessage.innerHTML = "Process is completed Successfully";
}
}
var amountLoaded = 0;
progressBarSim(amountLoaded);
</script>
</body>
```

<time>

This element is intended to be used presenting dates and times in a machine readable format. This can be helpful for user agents to offer any event scheduling for user's calendar. It is a paired tag.

Syntax:<time>......................</time>

Attributes:

Attribute	Value	Description
datetime	datetime	Gives the date/time being specified.

NOTE: The <time> tag does not render as anything special in any of the major browsers.

Example

```
<!doctype html>
<body>
We arrived at <time>09:00</time>
```

```
</body>
```

Example

```
<!doctype html>
<body>
I have an appointment with doctor on date
<time datetime="2017-09-02"> day</time>.
</body>
```

HTML5 Webforms

A Webform (HTML form) on a web page allows a user to enter data that is sent to a server for processing. These forms contains checkboxes, radio buttons, or text fields. Webforms are defined in formal programming languages such as HTML, Perl, Php, Java or .NET.

Web 1.0 was an early stage of the conceptual evolution of the World Wide Web, centered around a top-down approach to the use of the web and its user interface.

Web 2.0 was coined in 1999 to describe web sites that use technology beyond the static pages of earlier web sites. web 2.0 suggests a new version of the World Wide Web, it does not refer to an update to any technical specification. Web 2.0 include social networking sites, blogs, wikis, video sharing sites, hosted services, web applications, Web 2.0 offers all users the same freedom to contribute.

Key features of Web 2.0
Free Classification of Information
Rich User Experience
User as a Contributor
Long Tail
User Participation
Basic Trust
Dispersion.

New <input> Types in HTML5 (Web forms 2.0) (Advanced Forms)

Web Forms 2.0 has been integrated into HTML5. These features were originally part of a WHATWG specification called Web Forms 2.0, based upon existing work at the W3C. That specification has now been rolled into HTML5.

HTML5 has several new input types for forms. These new features allow better input control and validation. HTML5 defines a variety of new input types: sliders, number spinners, popup calendars, color choosers, autocompleting suggest boxes, and more..!!

HTML5 defines 13 new values for the type attribute of the HTML <input> element (search, tel, url, email, datetime, date, month, week, time, datetime-local, number, range, and color). In that 6 are date pickers and 7 are new text types.

1. color(color chooser)
2. date(popup calendar)
3. datetime(datetime chooser)
4. datetime-local(datetime chooser)
5. email(Email Entry)
6. month(month chooser)
7. number(spinner)
8. range(slider)
9. search(Search Query Input)
10. tel(Telephone Input)
11. time(TimeSelector)
12. url(URL Entry)
13. week(WeekChooser)

Note:
All major browsers support all the new input types.If they are not supported, they will behave as regular text fields.

Date pickers:
1. date
2. datetime
3. datetime-local
4. month
5. time
6. week

Input Type: date
The date type allows the user to select a date. A Date and time field can be easily found in many web forms. Typical applications are like ticket booking, appointment booking, ordering food Items and etc.

Syntax: Input type=name

Example:
```
<body>
<form>
<input type='date'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 Date
</title>
</head>
<body>
<form action="nit.html">
  My Birthday ON: <input type="date"
  name="birday" />
<input type="submit" value="DisplayDate" />
</form>
</body>
</html>
```

Input Type: datetime

The datetime type allows the user to select a date and time (with time zone). You can choose date and time with time zone. Input value is represented in UTC/GMT time.

Syntax: Input type=name

Example:

```
<body>
<form>
<input type='datetime'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 datetime
</title>
</head>
<body>
<form action="nit.html">
```

```
  MyBirthDay (date and time):
<input type="datetime" name="bdaytime" />
<input type="submit" value="Login"/>
</form>
</body>
</html>
```

Input Type: datetime-local
The datetime-local type allows the user to select a date and time (no time zone).

Syntax: Input type=name

Example:

```
<body>
<form>
<input type='datetime-local'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 datetime-local
</title>
</head>
<body>
<form action="nit.html">
Birthday (date and time):
<input type="datetime-local" name="bdaytime" />
<input type="submit" value="Login"/>
</form>
</body>
</html>
```

Input Type: month
The month type allows the user to select a month and year.
Syntax: Input type=name

Example:

```
<body>
```

```
<form>
<input type='month'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 month
</title>
</head>
<body>
<form action="nit.html">
 MyBirthday (month and year):
<input type="month" name="bdaymonth" />
<input type="submit" value="ClickMe"/>
</form>
</body>
</html>
```

Input Type: time

The time type allows the user to select a time. The time will be collected with the hour, minutes, seconds, and fractions of seconds. No timezone will be set.

Syntax:Input type=name

Example:

```
<body>
<form>
<input type='time'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 time
```

```
</title>
</head>
<body>
<form action="nit.html">
 Select a time:
<input type="time" name="usr_time" />
<input type="submit" value="Login"/>
</form>
</body>
</html>
```

Input Type: week
The week type allows the user to select a week and year. The week will be collected without any timezone information.

Syntax:Input type=name

Example:

```
<body>
<form>
<input type='week'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 week
</title>
</head>
<body>
<form action="nit.html">
  Select a WeekNumber:
<input type="week" name="yr_week" />
<input type="submit" value="ClickMe" />
</form>
</body>
</html>
```

HTML5&CSS3

1. color(color chooser)
2. email(Email Entry)
3. number(spinner)
4. range(slider)
5. search(Search Query Input)
6. tel(Telephone Input)
7. url(URL Entry)

Input Type: color
The color type is used for input fields that should contain a color. With Color input type, you no longer need a complex Javascript color picker.

Syntax: Input type=name

Example:
```
<body>
<form>
<input type='color'>
</form>
</body>
```

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 color-InputType
</title>
</head>
<body>
<form action="nit.html">
 Select your favorite color:
<input type="color" name="favcolor" /><br />
<input type="Submit" value="ClickMe" />
</form>
</body>
</html>
```

Input Type: email
The email type is used for input fields that should contain an e-mail address. The email INPUT tag gives a way to request email addresses in your web form.

Syntax: Input type=name

Example

```
<body>
<form>
<input type='email'>
</form>
</body>
```

HTML5 <input> required Attribute

The required attribute is a boolean attribute. When present, it specifies that an input field must be filled out before submitting the form.

Note: The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

Syntax

```
<input required="required" />
or
<input required>
or
<input required="">
```

Note: The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

Example:

```
<body>
<form action="nit.html">
  E-mail: <input type="email" name="email" required>
  <input type="submit">
</form>
</body>
```

Input Type: number

The number type is used for input fields that should contain a numeric value.

Syntax: Input type=name

Attributes

Use the following attributes to specify restrictions:

max - Specifies the maximum value allowed

min - Specifies the minimum value allowed

step - Specifies the legal number intervals
value - Specifies the default value

Example:

```
<!DOCTYPE html>
<body>
<form>
<input type='number'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 number-InputType
</title>
</head>
<body>
<form action="nit.html">
Quantity (between 1 and 8):
<input type="number" name="quantity" min="1" max="8" />
<input type="submit"  value="Select"/>
</form>
</body>
</html>
```

Example(step)

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 number-InputType
</title>
</head>
<body>
<form action="nit.html">
  <input type="number" name="points" step="3" />
  <input type="submit" value="ClickMe"/>
</form>
</body>
```

```
</html>
```

Input Type: range
The range type is used for input fields that should contain a value from a range of numbers. You can also set restrictions on what numbers are accepted.

Syntax:Input type=name

Attributes
Use the following attributes to specify restrictions:
1. max - Specifies the maximum value allowed
2. min - Specifies the minimum value allowed
3. step - Specifies the legal number intervals
4. value - Specifies the default value

Example:
The step attribute specifies the legal number intervals for an <input> element.
Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.

Note: The step attribute works with the following input types:
number, range, date, datetime, datetime-local, month, time and week

Syntax
```
<input step="number" />
```

Example:
```
<body>
<form>
<input type='range'>
</form>
</body>
```

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 rangge-InputType
</title>
</head>
<body>
<form action="nit.html">
```

```
Points:
<input type="range" name="points" min="1" max="10" />
<input type="submit" value="Login"/>
</form>
</body>
</html>
```

Example:(step)

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 number-InputType
</title>
</head>
<body>
<form action="nit.html">
<input type="range" name="points" step="3"/>
<input type="submit"  value="Login"/>
</form>
</body>
</html>
```

Input Type: search

In HTML5, we can define a textbox as search box instead of a normal textbox. Notice, there is a blue "cross" sign appears in the textbox when,you input something in the search box, when you click on the "cross", your input string will be clear and you can start to type a new string.

Syntax:

Input type=name

Example:

```
<body>
<form>
<input type='search'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
```

```
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 search-InputType
</title>
</head>
<body>
<form action="nit.html">
 Search Google:
<input type="search" name="googlesearch" />
<br />
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

Input Type: tel

Input fields that accept phone numbers use the "tel" type. Due to inherent variances in phone number formats, the tel input type does not conform to any specific pattern. The main advantage to using this type of field then is to optimize the keyboard on mobile devices.

There is a new attribute that can help you to enforce your preferred format. It's called the pattern element. When you include it, the browser will validate the field contents against the Regular Expression assigned to the pattern attribute.

Syntax:Input type=name

Example:

```
<body>
<form>
<input type='tel'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 tel-InputType
</title>
```

```
</head>
<body>
<form action="nit.html">
Telephone:
<input type="tel" name="usrtel" />
<br />
<input type="submit"  value="ClickMe"/>
</form>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 tel-InputType
</title>
</head>
<body>
<form action="nit.html">
<input type='tel'
pattern='[\+]\d{2}[\(]\d{2}[\)]\d{4}[\-]\d{4}'
title='Phone Number (Format: +99(99)9999-9999)'>
 <input type="submit"  value="ClickMe"/>
</form>
</body>
</html>
```

Input Type: url

The url type is used for input fields that should contain a URL address. The value of the url field is automatically validated when the form is submitted.

Web URLs are also inherently difficult to validate. At least they were, until HTML5 came along with the new 'url' input type. Browsers will still treat the field as a regular text box. mobile devices to replace the space bar with a period, a forward slash, and a ".com" virtual keys.

Syntax:Input type=name

Example:

```
<body>
```

```
<form>
<input type='url'>
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 url-InputType
</title>
</head>
<body>
<form action="nit.html">
Add your homepage:
<input type="url" name="homepage" /><br/>
<input type="submit" value="ClickMe"/>
</form>
</body>
</html>
```

Example:

```
<body>
<form action="nit.html">
  URL/URI: <input type="url" name="myurl" required>
  <input type="submit" value="Validate">
</form>
</body>
```

HTML5 New Form Attributes
HTML5 has several new attributes for <form> and <input>.

New attributes for <form>:
1. autocomplete
2. novalidate

<form> / <input> autocomplete Attribute
The autocomplete attribute specifies whether a form or input field should have autocomplete on or off. When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

Syntax
```
<form autocomplete="on/off">
```

Note:
The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New Form Attributes
</title>
</head>
<body>
<form action="nit.html" method="get" autocomplete="off">
  First name:<br/>
<input type="text" name="fname" /><br>
  E-mail: <br/>
<input type="email" name="email" /><br>
  <input type="submit" />
</form>
</body>
</html>
```

<form> novalidate Attribute:
The novalidate attribute is a boolean attribute. When present, it specifies that the form-data (input) should not be validated when submitted.

Syntax
<form novalidate="novalidate">

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New Form Attributes
</title>
</head>
```

```
<body>
<form action="nit.html"  novalidate="novalidate">
First name:<br/>
<input type="text" name="fname"  required="required"/> <br/>
E-mail: <br/>
<input type="email" name="ueid" required="required"> <br/>
<input type="submit" value='Validate'>
</form>
</body>
</html>
```

New attributes for <input>:
1. placeholder (TextFieldswith TemporaryHints)
2. autofocus
3. required (TextFields with Required Values[non-empty])
4. autocomplete
5. form
6. formaction
7. formenctype
8. formmethod
9. formnovalidate
10. formtarget
11. height and width
12. list ((autocompleting suggest box))
13. min and max
14. multiple
15. pattern (regexp)(Validating TextFields)
16. step
17. spellcheck
18 contenteditable
19 accesskey

HTML5 <input> placeholder Attribute
The placeholder attribute specifies a short hint that describes the expected value of an input field The hint is displayed in the input field when it is empty, and disappears when the field gets focus.

Note:
The placeholder attribute works with the following input types: text, search, url, tel, email, and password. The placeholder attribute is new in HTML5.

Syntax
```
<input placeholder="text/hint" />
```

Attribute Values

Value	Description
text	Specifies a short hint that describes the expected value of the input field

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html">
<fieldset>
<legend align="center">User Login Form...!!</legend>
<input type="text" name="fname" placeholder="First name" /><br/>
<input type="text" name="lname" placeholder="Last name" /><br/>
<input type="password" name="pwd" placeholder="Password" /><br/>
<input type="submit" value="Login" />
</fieldset>
</form>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Working with table
New input attributes
</title>
</head>
<body>
<table bgcolor="lightblue" height="10%" width="250">
<tr><td>User Name:</td><td><Input placeholder="UName"></td></tr>
<tr><td>Password:</td><td><Input type="password" placeholder="Password"></td></tr
<tr><td>&nbsp</td><td><Input type="button" value=Login ></td></tr>
</table>
```

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

HTML5 <input> autofocus Attribute
The autofocus attribute is a boolean attribute. When present, it specifies that an <input> element should automatically get focus when the page loads.

Syntax
```
<input autofocus="autofocus" />
or
<input autofocus>
or
<input autofocus="">
```

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html">
 First name:<br/>
<input type="text" name="fname" autofocus="autofocus"> <br>
Last name: <br/>
<input type="text" name="lname"><br>
<input type="submit"  value="NextPage"/>
</form>
</body>
</html>
```

HTML5 <input> required Attribute
The required attribute is a boolean attribute. When present, it specifies that an input field must be filled out before submitting the form.

Note:
The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

Syntax

```
<input required="required" />
or
<input required>
or
<input required="">
```

Note:
The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form  action="nit.html" id="form1" name="Myform">
<label>What is your favorite movie:</label><br/>
<input name="movie" type="text" required="required"/> <br/>
<input type="submit" value="Login"/>
</form>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html>
<head>
   <title>Required Attribute</title>
</head>
<body>
   <form id="myform">
      <label>Name:</label>&nbsp;<input type="text" id="name" required="true" /><br/>
      <label>MyCar:</label>&nbsp;<input type="text" id="car" required="true" /><br/>
      <br/>
      <input type="submit" id="btnsubmit" value="Submit!" />
   </form>
</body>
</html>
```

<input> autocomplete Attribute
The autocomplete attribute specifies whether or not an input field should have autocomplete enabled. Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values.

Note:
The autocomplete attribute works with the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

Syntax
```
<input autocomplete="on|off" />
```

Attribute Values

Value	Description
on	Default. Specifies that autocomplete is on (enabled)
off	Specifies that autocomplete is off (disabled)

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New Input Attributes
</title>
</head>
<body>
<form action="nit.html" autocomplete="off">
  First name:<br/>
<input type="text" name="fname" /><br/>
  Last name: <br/>
<input type="text" name="lname" /><br/>
  E-mail: <br/>
<input type="email" name="email" autocomplete="on"/> <br/>
<input type="submit"  value="Login"/>
</form>
</body>
</html>
```

<input> form Attribute
The form attribute specifies one or more forms an <input> element belongs to.

Syntax

```
<input form="id/name" />
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html" id="form1">
First name: <br/>
<input type="text" name="fname"><br/>
<input type="submit" value="Submit">
</form>
Last name: <br/>
<input type="text" name="lname" form="form1">
</body>
</html>
```

<input> formaction Attribute

The formaction attribute specifies the URL of a file that will process the input control when the form is submitted. The formaction attribute overrides the action attribute of the <form> element.

Note:

This attribute is used with the following input types

1."submit"

2. "image"

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
```

```
<body>
<form action="nit.html">
  First name: <br/>
<input type="text" name="fname"><br>
  Last name: <br/>
<input type="text" name="lname"><br>
<input type="submit" value="@form">
<input type="submit" formaction="html5.png" value="@input" >
</form>
</body>
</html>
```

<input> formenctype Attribute

The formenctype attribute specifies how the form-data should be encoded when submitting it to the server (only for forms with method="post") The formenctype attribute overrides the enctype attribute of the <form> element.

Note: It is used with type="submit" and type="image".

Syntax:

```
<input type formenctype=name>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html" method="get">
First name: <br/>
<input type="text" name="fname"><br/>
Password: <br/>
<input type="password" name="pwd"><br/>
<input type="submit" value="G@form">
<input type="submit" formenctype="form-data"
value="P@input">
</form>
</body>
</html>
```

<input> formmethod Attribute

The formmethod attribute defines the HTTP method for sending form-data to the action URL. The formmethod attribute overrides the method attribute of the <form> element.

Note:It can be used with type="submit" and type="image".

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html" method="get">
First name: <br/>
<input type="text" name="fname"><br/>
Last name: <br/>
<input type="text" name="lname"><br/>
Password: <br/>
<input type="password" name="pwd"><br/>
<input type="submit" value="@Form">
<input type="submit" formmethod="post" formaction="goodmorning.gif"
value="@Input">
</form>
</body>
</html>
```

Example2

```
<!doctype html>
<body>
<form action="nit.html">
 <input type="text" placeholder="User Name" name="fname"><br/>
<input type="password" placeholder="Password"  name="lname"><br/>
<input type="image" src="html5.png" alt="Submit" width="55" height="30" title='submit
using GET'><br/>
<input type="submit" formmethod="post" formaction="nit1.html" value="Submit using
POST">
</form>
</body>
```

<input> formnovalidate Attribute
The novalidate attribute is a boolean attribute. When present, it specifies that the <input> element should not be validated when submitted. The formnovalidate attribute overrides the novalidate attribute of the <form> element.

Note: It can be used with type="submit"

Syntax:
<input type formnovalidate='on|off'>

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html">
E-mail: <br/>
<input type="email" name="userid" required="required"><br>
User Name: <br/>
<input type="text" name="user" required="required"><br>
<input type="submit" value="@YValidate">
<input type="submit" formnovalidate="formnovalidate" value="@NValidate">
</form>
</body>
</html>
```

<input> formtarget Attribute
The formtarget attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form. The formtarget attribute overrides the target attribute of the <form> element.

Note: It can be used with type="submit" and type="image".

Syntax:
<input type formtarget='target'>

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html">
  First name: <br/>
<input type="text" name="fname"><br>
  Last name: <br/>
<input type="text" name="lname"><br>
<input type="submit" value="@SameTW">
<input type="submit" formtarget="_blank" value="@NewTW">
</form>
</body>
</html>
```

<input> height and width Attributes

The height and width attributes specify the height and width of an <input> element.

Note:

These attributes are only used with <input type="image">.

Syntax:

```
<input type='image' height='' " width="">
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html">
  First name: <br/>
<input type="text" name="fname"><br/>
```

```
  Password: <br/>
<input type="password" name="pwd"><br/>
<input type="image" src="html5.png" width="25px" height="15px">
</form>
</body>
</html>
```

<input> list Attribute

The list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.

Syntax: <input list="Name">

Example

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html" method="get">
<input list="browsers" name="browser">
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
<input type="submit"></form>
</body>
</html>
```

<input> min and max Attributes

The min and max attributes specify the minimum and maximum value for an <input> element.

Note:

The min and max attributes works with the following input types: number, range, date, datetime, datetime-local, month, time and week.

Syntax:
```
<input type min=value max=value>
```

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Webforms 2.0 number-InputType
</title>
</head>
<body>
<form action="nit.html">
Quantity (between 1 and 8):
<input type="number" name="quantity" min="1" max="8" />
<input type="submit"  value="Select"/>
</form>
</body>
</html>
```

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html">
 Enter a date before 1980-01-01:
 <input type="date" name="bday" max="1979-12-31"><br>
 Enter a date after 2000-01-01:
 <input type="date" name="bday" min="2000-01-02"><br>
 Quantity (between 1 and 5):
  <input type="number" name="quantity" min="1" max="5"><br>
  <input type="submit">
  </form>
  </body>
  </html>
```

<input> multiple Attribute
The multiple attribute is a boolean attribute. When present, it specifies that the user is allowed to enter more than one value in the <input> element.

Note: It supports email, and file input types.

Syntax:
```
<input type='file' multiple='on|off'>
```

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html">
  Select images: <input type="file" name="img" multiple="multiple">
  <input type="submit" value='Login'>
</form>
<p>Try selecting more than one file when browsing for files.</p>
</body>
</html>
```

<input> pattern Attribute
The pattern attribute specifies a regular expression that the <input> element's value is checked against.

Note:
The pattern attribute works with the following input types: text, search, url, tel, email, and password.

Syntax:
```
<input type pattern=RegExp>
```

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
```

```
<meta charset='utf-8'>
<title>
Webforms 2.0 tel-InputType
</title>
</head>
<body>
<form action="nit.html">
<input type='tel'
pattern='[\+]\d{2}[\(]\d{2}[\)]\d{4}[\-]\d{4}'
title='Phone Number (Format: +99(99)9999-9999)'>
 <input type="submit"  value="ClickMe"/>
</form>
</body>
</html>
```

Example:

```
<!doctype html>
<body>
<form action='nit1.html' name="Myform" id="form1">
<label style='color:blue;font-family:tahoma'>Enter Valid Country Code <b
style='color:red'>*</b></label><br/>
<input type='text' placeholder="OnlyCountryCode" title="SorryInvalidCountryCode"
pattern="[a-zA-Z]{2}" required="required"  name='ccode' id='txt1' />
<br/>
<input type='submit' value="NextPage" />
<input type='reset' value="Cancel" />
</form>
</body>
```

<input> step Attribute

The step attribute specifies the legal number intervals for an <input> element.

Note:

The step attribute works with the following input types: number, range, date, datetime, datetime-local, month, time and week.

Syntax:

```
<input type=name step=value>
```

Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.

Example:

```
<!DOCTYPE html>
```

```
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
New input attributes
</title>
</head>
<body>
<form action="nit.html">
  <input type="number" name="points" step="3">
  <input type="submit" value='Login'>
</form>
</body>
</html>
```

HTML spellcheck Attribute
The spellcheck attribute specifies whether the element is to have its spelling and grammar checked or not.

The following can be spellchecked:
- Text values in input elements (not password)
- Text in <textarea> elements
- Text in editable elements

Syntax
```
<element spellcheck="true|false">
```

Attribute Values

Value	Description
true	The element is to have its spelling and grammar checked
false	The element is not to be checked (default)

Example:
```
  <body>
<input type="text" spellcheck="true" > <br/>
<p><textarea spellcheck="true">Text area element </textarea></p>
</body>
```

contenteditable:
Using this attribute we can modify the content directly on the web Page.

Syntax
```
<element contenteditable="true|false">
```

Attribute Values

Value	**Description**
true	The element content is able to modify
false	The element content is unable to modify. (default)

Example:

```
<!DOCTYPE html>
<html>
<body>
<p contenteditable="true" spellcheck="true">This is a praggagraph. It is editable. Try to change the text.</p>
<p contenteditable="false" spellcheck="false">This is a paaragraph. It is not editable. Try to change the text.</p>
</body>
</html>
```

HTML accesskey Attribute:

Accesskeys allow easier navigation by assigning a keyboard shortcut to a link (which will usually gain focus when the user presses 'Alt' or 'Ctrl' + the accesskey). For users who do not use pointing devices, this is a much quicker and easier way to navigate than tabbing through links.

The accesskey attribute specifies a shortcut key to activate/focus an element.

Syntax

```
<element accesskey="character">
```

Attribute Values

Value	Description
character	Specifies the shortcut key to activate/focus the element

Example:

```
<a href="nit.html" accesskey="s">Some page</a>
```

Example:

```
<html>
<body>
<a href="http://www.nareshit.com" accesskey="h">HTML tutorial</a><br>
<a href="http://www.w3.com" accesskey="c">CSS tutorial</a>
<p>Note:Use Alt + accessKey (or Shift + Alt + accessKey) to access the element with the specified access key.</p>
</body>
</html>
```

HTML5&CSS3

HTML5 Forms--> New Form Elements
HTML5 offers more form elements, with more functionality, HTML5 has the following new form elements:
 <datalist>
 <keygen>
 <output>

<datalist>
The <datalist> tag is used to list a set of data to be used in a list of options like autocomplete feature used in forms. It enables you to provide a list of predefined options to the user as they input data. It is a paired tag.

Syntax:
<datalist>.....................</datalist>

Attributes
data--> Specifies an XML file that can be used to prefill the datalist.

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
 <label>
  Enter your favorite color:<br />
  <input type="text" name="favcolor" list="colors">
  <datalist id="colors">
   <option value="Green">
   <option value="Blue">
   <option value="White">
  <option value="Maroon">
  <option value="Lime">
  </datalist>
 </label><br />
</p>
</body>
</html>
```

Example:
```
<!DOCTYPE html>
```

```
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<form action="nit.html" method="get">
<input list="browsers" name="browser" />
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
<input type="submit" value="ClickMe"/>
</form>
</body>
</html>
```

<keygen>

Generate keys to authenticate users.The purpose of the <keygen> element is to provide a secure way to authenticate users. The keygen tag is used to key generator for a form.The private key is stored broswer and the public key is sent to the server when the form is click or submitted. It is an empty tag.

Syntax: <keygen>

Attributes

Attribute	**Value**	**Description**
autofocus	autofocus	<keygen> element should automatically get focus when the page loads.
challenge	challenge	<keygen> element should be challenged when submitted.
disabled	disabled	Specifies that a <keygen>element should be disabled.
name	name	Defines a name for the <keygen>element

Example:

```
<body>
<form action="nit.html" method="post">
Username: <br/>
<input type="text" name="usr" id="usr1" /><br/>
Password: <br/>
<input type="password" name="pwd" id="pwd1" /><br/>
```

```
Encryption Key: <br/>
<keygen name="enname" id="enname1" challenge="challenge"/><br/>
<input type="submit" value="NextPage" />
</form>
</body>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<form action="nit.html" method="post">
<input type="text" placeholder='User Name'/>
<br/>
<input type='password' placeholder="Password">
<br/>
Encryption:<keygen name="security"/>
<br/>
<input type="submit"  value="Submit"/>
</form>
</body>
</html>
```

When the form is submitted, the private key gets stored in the local keystore, and the public key is sent to the server.

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<form action="nit.html" method="get">
 Username: <br/>
<input type="text" name="uname" id="user1" /> <br/>
 Encryption Key: <br/>
<keygen name="encrypt" id="encrypt1" challenge="challenge" /><br/>
<input type="submit" value="Generate" />
</form>
</body>
</html>
```

In the above form User Name is a login name and encryption field will to encrypt a password.In the Encryption keys is having two types first one is "High Grade" and second one is "Medium Grade".when form will submit the keys are stored on server and client side.

<output>
HTML <output> tag is used for representing the result of a calculation, The output tag is mainly used to calculate the result example add,subtract,division and multiplication through form. It is a container tag.

Note:
The <output> tag is supported in all major browsers, except Internet Explorer.

Syntax: <output>.............................</output>

Attributes

Attribute	**Value**	**Description**
for	element_id	Spe. the relationship between the result of the Calculation.
form	form_id	Spe. one or more forms the output element belongs to
name	name	Specifies a name for the output element

Example
```
<!DOCTYPE html>
<html>
<body>
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">Number
<input type="range" name="a" value="50" />Number
+<input type="number" name="b" value="50" />
=<output name="x" for="a b"></output>
</form>
</body>
</html>
```

JavaScript parseInt() Function:
The parseInt() function parses a string and returns an integer.

Syntax
parseInt(string)

Parameter	**Description**
string	Required. The string to be parsed

JavaScript parseFloat() Function:
The parseFloat() function parses a string and returns a floating value.

Syntax
parseFloat(string)

Parameter	**Description**
string	Required. The string to be parsed

HTML Multimedia
Multimedia on the web is sound, music, videos, and animations. Modern web browsers have support for many multimedia formats.

History of multimedia in the internet
First version of RealPlayer in 1995
Video support for Flash 8 in 2005
Windows Media Player and iTunes surpassed RealPlayer in 2007

What is Multimedia?
Multimedia comes in many different formats. It can be almost anything you can hear or see. Modern Web pages have often embedded multimedia elements, and modern browsers have support for various multimedia formats.

Examples:
Pictures, music, sound, videos, records, films, animations, and more.

The plugin era
Plugins proliferated:
1991: Apple Quicktime (.mov)
1994: MIDI (background music)
1995: RealAudio (.ra, .ram)
1997: RealVideo (H.263)
1998: Windows Media Player
1999: Quicktime for Windows (.mov, .mp3, .mp4, .swf)
2002: Macromedia Shockwave Flash (.swf)
2007: Microsoft Silverlight (flash, vid, etc.)

Browser Support
The first Internet browsers had support for text only, and even the text support was limited to a single font in a single color. Then came browsers with support for colors, fonts and text styles, and support for pictures was added.

The support for sounds, animations, and videos is handled in different ways by various browsers. Some multimedia elements is supported, and some requires an extra helper program (a plug-in).

Multimedia Formats:(Video formats)
Multimedia files also have their own formats with different extensions like: .swf, .wav, .mp3, and .mp4. MP4 is the new and upcoming format for internet video. It is supported by YouTube, Flash players and HTML5.

AVI(.avi)-->AVI (Audio Video Interleave) was developed by Microsoft. AVI is supported by all computers running Windows, and by the most popular web browsers.

WMV (.wmv)-->WMV (Windows Media Video) was developed by Microsoft. WMV is a common format on the Internet.

MPEG(.mpg or .mpeg)-->The MPEG (Moving Pictures Expert Group) format is the most popular format on the Internet.

Flash(.swf or .flv)--> Flash was developed by Macromedia.

MP4 (.mp4 Mpeg-4 (MP4) is the new format for the internet.

Multimedia Formats:(Audio formats)
MIDI (.mid or .midi)-->MIDI (Musical Instrument Digital Interface) is a format for electronic music devices like synthesizers and PC sound cards.

MP3(.mp3)-->MP3 files are actually the sound part of MPEG files.

WAV (.wav)-->WAVE (more known as WAV) was developed by IBM and Microsoft. WAVs are compatible with Windows, Macintosh, and Linux operating systems.

WMA(.wma)-->WMA (Windows Media Audio)

HTML4 Multimedia Tags:
HTML <object> Tag
The <object> tag defines an embedded object within an HTML document. Use this element to embed multimedia (like audio, video, Java applets, ActiveX, PDF, and Flash) in your web pages. It is a paired tag

Syntax: <object>...............</object>

Example:
<html>

```
<body>
<object data="Windows XP Shutdown.wav">
<param name="autoplay" value="true" />
</object>
 </body>
</html>
```

Example:

```
<!DOCTYPE html>
<object data="Windows XP Shutdown.wav" type="audio/x-mplayer2" width="320"
height="240">
<param name="src" value="music.wav">
<param name="autoplay" value="false">
<param name="autoStart" value="0">
Hear the sound : <a href="Windows XP Shutdown.wav">music</a>
</object>
```

<param> tag:
You can use the <param> tag to pass parameters to plugins that have been embedded with the <object> tag. It is a Non-Paired Tag..

Syntax: <param>

Example:

```
<html>
<body>
<object data="Windows XP Shutdown.wav">
<param name="autoplay" value="true" />
</object>
 </body>
</html>
```

<embed>
The HTML <embed> tag is used for embedding an external application or interactive content into an HTML document. It is non-paired tag.

Syntax: <embed>

Attributes

Attribute	Value	Description
height	pixels	Specifies the height of the embedded content
width	pixels	Specifies the width of the embedded content
src	URL	Specifies the address of the external file to embed

type	MIME_type	Specifies the MIME type of the embedded content

YouTube in HTML5

YouTube videos can be included in HTML documents, and Google offers the code to do so right on the same page as the video itself!

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<embed  src="http://www.youtube.com/v/opVb89Cmrtk&hl=en&fs=1"  type="application/x-shockwave-flash"  width="425" height="344" >
</body>
</html>
```

Example:

```
<embed src="http://www.youtube.com/v/BHtGcZ9XqjY"
    type="application/x-shockwave-flash"
     allowscriptaccess="always"
     allowfullscreen="true" width="480" height="385"/>
```

HTML <bgsound> Tag

The HTML <bgsound> tag is used to play a soundtrack in the background. This tag is for Internet Explorer only.

Specific Attributes

The HTML <bgsound> tag also supports following additional attributes:

Attribute	Value	Description
loop	number	Lets you replay a background soundtrack a certain number of times.
src	URL	Specifies the path of the sound file.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML bgsound Tag</title>
</head>
<body>
<bgsound src="/html/yourfile.mdi"/>
<p>This does create any result on the screen but it plays sound file in the background.</p>
```

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

<source>

The HTML <source> tag is used to specify multiple media resources on media elements (such as <audio> and <video>).

Syntax: <source>....................</source>

Attributes

Attribute	Value	Description
media	media_query	Specifies the type of media resource
src	URL	Specifies the URL of the media file
type	MIME_type	Specifies the MIME type of the media resource.

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<audio controls="controls">
<source src="good_enough.mp3" type="audio/mpeg" />
Your browser does not support the audio element.
</audio>
</body>
</html>
```

Example

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Working with Source Tag
</title>
</head>
<body>
<video controls='controls'>
<source src="oceans.mp4" type="video/mp4" />
Your browser does not support the Video element.
</video>
```

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

HTML5-MULTIMEDIA (new standard for media content)
(playing video and audio is easier than ever)

1. The <audio> and <video> elements embed and allow the manipution of new multimedia content.
2. Track and WebVTT
The <track> element allows subtitles and chapters. WebVTT is a text track format.

The <audio> Tag
The HTML <audio> tag is used to specify audio on an HTML document. It Enables audio playback without plugins, The audio can be controlled with native or customized controls, Audio files can be prefetched or downloaded on-demand. It is a paired tag.

Note:
Any text inside the between <audio> and </audio> will be displayed in browsers that do not support audio.

Syntax: <audio>....................</audio>

Attributes

Attribute	**Value**	**Description**
autoplay	autoplay	Specifies that the audio will start playing
controls	controls	Specifies that audio controls should be displayed
loop	loop	Specifies that the audio will start over again
src	URL/path of audio	Specifies the URL of the audio file

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<audio controls="controls">
  <source src="song.ogg"  type="audio/ogg" />
   Your browser does not support the audio element.
</audio>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
  <body>
    <audio src="song.ogg" controls="controls" autoplay="autoplay" loop="loop">
      <p>Your browser does not support the audio playback. Please upgrade to a modern
          browser.</p>
    </audio>
  </body>
</html>
```

Example: (from Web)

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
  <body>
    <audio src="http://www.nareshit.com/html5/demo/demo.mp3"          controls="controls"
autoplay="autoplay" loop="loop">
<p>Your browser does not support the audio playback. Please upgrade to a modern
browser.</p>
    </audio>
  </body>
</html>
```

Example:(wav)

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
  <body>
    <audio src="Windows XP Shutdown.wav"   controls="controls"  autoplay="autoplay"
loop="loop">
      <p>Your browser does not support the audio playback. Please upgrade to a modern
browser.</p>
    </audio>
  </body>
</html>
```

Audio With JS

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<audio id="demo" src="good_enough.mp3" controls='controls'></audio>
<div>
  <button onclick="document.getElementById('demo').play()">Play the Audio</button>
  <button onclick="document.getElementById('demo').pause()">Pause the Audio</button>
  <button onclick="document.getElementById('demo').volume+=0.1">Increase Volume</button>
  <button onclick="document.getElementById('demo').volume-=0.1">Decrease Volume</button>
</div>
</body>
</html>
```

What is Ogg?
Ogg is a free, open container format maintained by the Xiph.Org Foundation. The creators of the Ogg format state that it is unrestricted by software patents and is designed to provide for efficient streaming and manipulation of high quality digital multimedia.

What is WebM?
The WebM Project is dedicated to developing a high-quality, open video format for the web that's freely available to everyone.

<video>
The HTML 5 <video> tag is used to specify video on an HTML document. It is a paired tag.

Syntax:
<video>.........................</video>

It can satisfy the following points:
1.Theora decoder not found
2.Video playback without plugins
3.Native or customized controls
4.No DRM(Digital Rights Management) support
5.No commonly agreed file format (more about this later)

Note:

Any text between the <video> and </video> tags will be displayed in browsers that do not support video.

Attributes:

Attribute	**Value**	**Description**
autoplay	autoplay	Specifies that the video will start playing
controls	controls	Specifies that video controls should be displayed
src	URL	Specifies the URL of the video file
width	pixels	Sets the width of the video player
height	pixels	Sets the height of the video player
loop	loop	Specifies that the video will start over again
muted	muted	Specifies that the audio output of the video should be muted
poster	URL	Spe. an image to be shown while the video is downloading, or until the user hits the play button
preload	auto metadata none	Spe.if and how the author thinks the video should be loaded when the page loads

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<video width="300" height="250" controls="controls" autoplay="autoplay">
  <source src="movie.mp4" type="video/mp4" />
<p>    OOPs Your browser does not support the video tag. </p>
</video>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<video width="300" height="250" controls="controls"
poster="http://nareshit.com/images?q=tbn:ANd9GcQUc6Vt3q5h6tcyn3229tN6u7fcG9Z0WXl
0fFYKUvnmj1GBNvpi">
```

```
  <source src="movie.mp4"  />
    Your browser does not support the video tag.
</video>
</body>
</html>
```

Video with JS

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<video id="demo" src="movie.mp4" controls='controls'></video>
<div>
  <button onclick="document.getElementById('demo').play()">Play the Video</button>
  <button onclick="document.getElementById('demo').pause()">Pause the Video</button>
  <button onclick="document.getElementById('demo').volume+=0.1">Increase
Volume</button>
  <button onclick="document.getElementById('demo').volume-=0.1">Decrease
Volume</button>
</div>
</body>
</html>
```

Playing Ogg videos using a Java applet

There's a Java applet called Cortado that you can use as a fallback to play Ogg videos in browsers that have Java support but don't support HTML5 video:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<video src="movie.ogg" controls width="320" height="240">
  <object type="application/x-java-applet" width="320" height="240">
     <param name="archive" value="cortado.jar">
     <param name="code" value="com.fluendo.player.Cortado.class">
     <param name="url" value="my_ogg_video.ogg">
     <p>You need to install Java to play this file.</p>
  </object>
</video>
```

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

HTML5 <track> Tag

The <track> tag specifies text tracks for media elements (<audio> and <video>). This element is used to specify subtitles, caption files or other files containing text, that should be visible when the media is playing.

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<video width="300" height="200" controls="controls">
 <source src="mov_bbb.ogg" type="video/ogg">
 <track src="subtitles_en.vtt" kind="subtitles" srclang="en"  label="English">
</video>
</body>
</html>
```

WebVTT

WebVTT is a format for displaying timed text tracks (e.g. subtitles) with the <track> element. The primary purpose of WebVTT files is to add subtitles to a <video>. WebVTT is a text based format. A WebVTT file must be encoded in UTF-8 format. Where you can use spaces you can also use tabs. The mime type of WebVTT is text/vtt.

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
</head>
<body>
<video src="pass-countdown.ogg" width="300" height="150" controls>
<track src="countdown_en.vtt" kind="subtitles" srclang="en" label="English">
<p>Browser does not support the HTML5 video element</p>
</video>
</body>
</html>
```

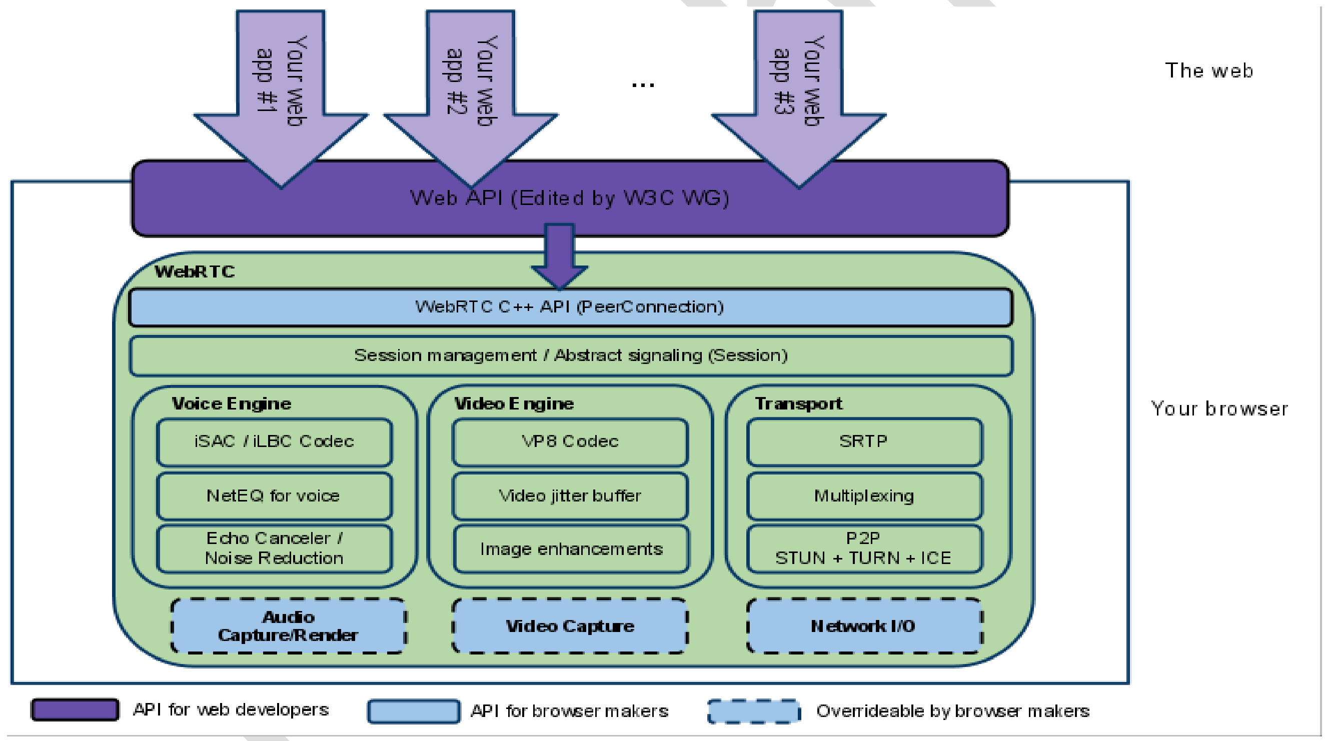
2. WebRTC (Real-time Communication Between Browsers)

HTML5&CSS3

WebRTC is a set of JavaScript APIs that enable peer-to-peer, realtime communication between web browsers. RTC stands for Real-Time Communication, allow to connect to other people and control videoconferencing directly in the browser, without the need for a plugin or an external application.

HISTORY OF WEBRTC:
Gmail video chat became popular in 2008, and in 2011 Google introduced Hangouts, which use the Google Talk service (as does Gmail). Google bought GIPS (Global Positioning System), a company which had developed many of the components required for RTC.

Real-time communication without plugins:
The RTC in WebRTC stands for Real-Time Communications, technology that enables audio/video streaming and data sharing between browser clients (peers). The goal of WebRTC is to enable applications such as voice calling, video chat and P2P file sharing without plugins.



KeyFeatures:
Media Streams: Access Users camera and Mic
PeerConnection:Easy Audio/Video calls
DataChannels:P2P Application data transfer.

MediaStreams:(Getting access to Audio and Video)

A MediaStream is a stream of audio and/or video data. When working locally, one can be obtained by calling getUserMedia. After a successful WebRTC connection is established, access to the remote browser's media stream will be available.

Example:

```
<!DOCTYPE html>
<html>
 <head><title>getUserMedia</title></head>
 <body>
  <video autoplay></video>
  <script>
   navigator.getUserMedia = navigator.getUserMedia || navigator.webkitGetUserMedia ||
navigator.mozGetUserMedia;

   var constraints = { audio: false, video: true };
   var video = document.querySelector("video");

   function successCallback(stream) {
    video.src = window.URL.createObjectURL(stream);
   }

   function errorCallback(error){
    console.log("getUserMedia error: ", error);
   }

   navigator.getUserMedia(constraints, successCallback, errorCallback);
  </script>
 </body>
</html>
```

PeerConnection (Live Audio and Video Sessions)

RTCPeerConnection is the WebRTC component that handles stable and efficient communication of streaming data between peers.

Keywords in WebRTC Architecture:

1 Your Web App
2 Web API
3 WebRTC Native C++ API
4 Transport / Session
5 RTP Stack
6 STUN/ICE
7 Session Management
8 VoiceEngine

9 iSAC
10 iLBC
11 NetEQ for Voice
12 Acoustic Echo Canceler (AEC)
13 Noise Reduction (NR)
14 VideoEngine
15 VP8
16 Video Jitter Buffer

Caller

Create a new RTCPeerConnection and add the stream from getUserMedia():

```
 pc1 = new webkitRTCPeerConnection(servers);
 pc1.addStream(localstream);
```

Create an offer and set it as the local description for pc1 and as the remote description for pc2. This can be done directly in the code without using signaling, because both caller and callee are on the same page:

```
pc1.createOffer(gotDescription1);
function gotDescription1(desc){
  pc1.setLocalDescription(desc);
  trace("Offer from pc1 \n" + desc.sdp);
  pc2.setRemoteDescription(desc);
  pc2.createAnswer(gotDescription2);
}
```

Callee

Create pc2 and, when the stream from pc1is added, display it in a video element:

```
pc2 = new webkitRTCPeerConnection(servers);
pc2.onaddstream = gotRemoteStream;
function gotRemoteStream(e)
{
vid2.src = URL.createObjectURL(e.stream);
}
```

3. Capturing Audio & Video in HTML5 or Using the Camera API (HTML Media Capture)

This specification extends the HTMLInputElement interface with a new capture attribute. The attribute enables content authors to give hints of preferred means to capture local media such as images, video, and sound, that is to be subsequently uploaded. Allows to use, manipulate and store an image from the computer's camera.

Capturing Points:

HTML5&CSS3

1. Your Application will have a local storage portion for pictures.
2. That local storage cab be filled directly from the device camera roll or by taking pictures.
3. The camera uses the devices native camera software.
4. Require a device with camera

Keyword	State	Capture control type
camera	Image Capture	A camera
camcorder	Video Capture	A video camera
microphone	Sound Capture	A sound recorder
filesystem	File Upload	A generic file picker

HTML Media Capture

HTML Media Capture was the DAP's(Digital Audio Player) first go at standardizing media capture on the web. It works by overloading the <input type="file"> and adding new values for the accept parameter.

Syntax:

Recording a video or audio is similar:

```
<input type="file" accept="image/*" capture="camera">
<input type="file" accept="video/*;capture=camcorder">
<input type="file" accept="audio/*;capture=microphone">
```

HTML5-GRAPHICS (2D and 3D Effects)

The Canvas Element
The canvas element uses JavaScript to make drawings on a web page.
For making graphics with a script
Canvas 3D (WebGL)
Inline SVG

```
<!--[if IE]>
<script type="text/javascript"
src="excanvas.js">
</script>
<![endif]-->
```

Animate your HTML5
with CSS3, SVG, Canvas and WebGL

The web has always been a visual medium, but a restricted one at best. With the addition of technologies like the canvas element, Web GL, and SVG images, In fact, there are many new features which deal with graphics on the web: 2D Canvas, WebGL, SVG, 3D CSS transforms, and SMIL (Synchronized Multimedia Integration Language)

HTML5&CSS3

BROWSER SUPPORT
"Graphics" is such a wide topic that there are several specifications being worked on. A comprehensive graphics required the following features:

PNG Alpha Transparency
Data URLs
CSS3 Colors
SVG (basic support)
Canvas (basic support)
Video element
CSS Transforms
Text API for Canvas
SVG in CSS Backgrounds
SVG effects for HTML elements
SVG SMIL animation (Synchronized Multimedia Integration Language)
SVG Fonts
SVG filters
CSS3 Animation
Inline SVG in HTML5
3D Canvas graphics / WebGL
CSS 3D Transforms etc...

HTML 5 <canvas> Tag
The HTML <canvas> tag is used for creating graphics on the fly. It can be used for rendering graphs, game graphics, or other visual images. You must use a script to actually draw the graphics.

<canvas> tag use to draw graphics using javascript getElementById("XXX").The <canvas> element is only a container for graphics. Canvas has several methods for drawing paths, boxes, circles, characters, and adding images.

HTML5 Canvas, including line and curve drawing, path drawing, shape drawing, gradients, patterns, images, and text. The HTML5 Canvas element is an HTML tag similar to the <div>, <a>, or <table> tag, with the exception that its contents are rendered with JavaScript. Access the canvas tag with JavaScript, create a context, and then utilize the HTML5 Canvas API to draw visualizations.

The HTML5 canvas element is used to create
1. Graphs and Charts
2. Animations
3. Games
4. Diagrams
5. Videos and Photo galleries

6. Special image effects
7. Drawing applications
8. User interface enhancements

Syntax: <canvas>........................</canvas>

Note:
Any text inside the <canvas> element will be displayed in browsers that does not support <canvas>.

Note:
Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas.

Attributes

Attribute	**Value**	**Description**
height	pixels	Specifies the height of the canvas
width	pixels	Specifies the width of the canvas

Global Attributes/Dynamic Attributes

Attributes	**Value**	**Description**
id	ID_Name	Declared unique id for the element.
class	Predefined_Class	Used in Cascading Style Sheet
style	style_Attributes	CSS code specify inline the HTML tag is presented
title	Title_Description	Display on the "tooltip" for your elements.

Event Attributes
<canvas> tag supported following Event attributes.:

Attributes	**Value**	**Description**
onfocus	"Java_script"	element gets focus on object when script to be run.
onblur	"Java_script"	element lose the focus on object when script to run
onabort	"Java_script"	element gets aborted on object when script torun
onchange	"Java_script"	element gets anytime change on object, script run
onclick	"Java_script"	clicked on object when script tobe run.

<canvas> Tag Contains the folloings features:

1. Create a Canvas
2. Draw Onto The Canvas With JavaScript
3. Canvas Coordinates
4. Canvas - Paths (lines, circles)
5. Canvas - Text

6. Canvas - Gradients
7. Canvas - Images

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="myCanvas" width="250" height="150" style="border:2px solid #0000FF;">
Your browser does not support the HTML5 canvas tag.
</canvas>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:2px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="#FFFF00";
ctx.fillRect(0,0,150,75);
</script>
</body>
</html>
```

Example explained:

1. find the <canvas> element: var c=document.getElementById("myCanvas");

2. Call its getContext() method (you must pass the string "2d" to the getContext() method):
var ctx=c.getContext("2d");
3. The getContext("2d") object is a built-in HTML5 object, with many properties and methods for drawing paths, boxes, circles, text, images, and more.
4. The next two lines draws a red rectangle:
ctx.fillStyle="#FF0000"; ctx.fillRect(0,0,150,75);
5. The fillStyle property can be a CSS color, a gradient, or a pattern. The default fillStyle is #000000 (black).
6. The fillRect(x,y,width,height) method draws a rectangle filled with the current fill style.

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="myCanvas">Your browser does not support the HTML5 canvas tag.</canvas>
<script type="text/javascript">
var c=document.getElementById('myCanvas');
var ctx=c.getContext('2d');
ctx.fillStyle='#FF0000';
ctx.fillRect(0,0,80,100);
</script>
</body>
</html>
```

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="CanvasEx"> </canvas>
<script type="text/javascript">
var canvas=document.getElementById('CanvasEx');
var ctx=canvas.getContext('2d');
ctx.fillStyle='#FFcc00';
ctx.fillRect(0,5,80,110);
ctx.fillStyle='#FF9900';
ctx.fillRect(25,0,120,50);
ctx.fillStyle='#FF00cc';
ctx.fillRect(100,35,120,60);
</script>
</body>
</html>
```

Canvas Coordinates
1. The canvas is a two-dimensional grid.
2. The upper-left corner of the canvas has coordinate (0,0)
3. The fillRect() method above had the parameters (0,0,150,75).
4. This means: Start at the upper-left corner (0,0) and draw a 150x75 pixels rectangle.

Canvas - Paths
To draw straight lines on a canvas, we will use the following two methods:
1. moveTo(x,y) defines the starting point of the line
2. lineTo(x,y) defines the ending point of the line
To actually draw the line, we must use one of the "ink" methods, like stroke().

Example:
HTML5 Canvas Line:

```
<script>
  context.beginPath();
  context.moveTo(x,y);
  context.lineTo(x,y);
  context.stroke();
</script>
```

Description:
The beginPath() defines a new drawing path.

The moveTo() method creates a new subpath for the given point. This point becomes the new context point. You can think of the moveTo() method as a way to position your drawing cursor.

The lineTo() method draws a line from the context point to the given point.

The stroke() method assigns a color to the line and makes it visible. Unless otherwise specified, the default stroke color is black.

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">
Your browser does not support the HTML5 canvas tag.</canvas>
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
</body>
</html>
```

Example:

```
<body>
<canvas width=200px height=100px style='border:2px solid #FF0000' id="MyCan">
<p style='color:red'>OOPs Your Browser not supporting Canvas feature Update and
Try....!</p>
<canvas>
<script type='text/javascript' language="javascript">
var c=document.getElementById("MyCan");
var ctx=c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.moveTo(200,0);
ctx.lineTo(0,100)
ctx.stroke();
</script>
</body>
```

Example:

```
<body>
<canvas width=200px height=100px style='border:2px solid #FF00FF' id='MyCan'>
```

```
<p>OOPs ....</p>
</canvas>
<script type='text/javascript'>
var c=document.getElementById('MyCan');
var ctx=c.getContext('2d');
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.moveTo(0,100);
ctx.lineTo(200,0);
ctx.moveTo(0,50);
ctx.lineTo(200,50);
ctx.moveTo(100,0);
ctx.lineTo(100,100);
ctx.stroke();
</script>
</body>
```

Example:

```
<!doctype html>
<body>
<canvas width="200px" height="100px" style="border:2px solid #FF0000" id="CanV">
<p style="color:red">OOPs Your Browser not supporting canvas feature update and Try...!!</p>
</canvas>
<script type='text/javascript' language="javascript">
var c=document.getElementById("CanV");
var ctx=c.getContext("2d");
ctx.moveTo(0,50);
ctx.lineTo(200,50);
ctx.moveTo(100,0);
ctx.lineTo(100,100);
ctx.moveTo(0,50);
ctx.lineTo(100,0);
ctx.moveTo(100,0);
ctx.lineTo(200,50);
ctx.moveTo(200,50);
ctx.lineTo(100,100);
ctx.moveTo(100,100);
ctx.lineTo(0,50);
ctx.stroke();
</script>
</body>
```

Example: arc(x,y,r,start,stop)--> Circle

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
</script>
</body>
</html>
```

Canvas - Text

To draw text on a canvas, the most important property and methods are:

font - defines the font properties for text
fillText(text,x,y) - Draws "filled" text on the canvas
strokeText(text,x,y) - Draws text on the canvas (no fill)
Using fillText():

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:2px solid #F3F3F3;">
</canvas>
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.font="30px Rosewood Std";
ctx.fillText("NareshiTech",10,50);
</script>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:2px solid #F3F3F3;">
</canvas>
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.font="30px Tahoma";
ctx.strokeText("NareshiTech",10,50);
</script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>Reflection Matrix</title>
 <script type="text/javascript">
      var can, ctx;
       function init() {
         can = document.getElementById("can");
         ctx = can.getContext("2d");
         ctx.fillStyle = "blue";
         ctx.font = "48pt Segoe Script";
         ctx.fillText("Reflection", 0, 100);
         ctx.setTransform(1, 0, 0, -1, 0, 0);
         ctx.fillStyle = "red";
```

```
        ctx.fillText("Reflection", 0, -100);
      }
   </script>
</head>
<body onload="init()">
   <canvas id="can" height="200" width="300">
   </canvas>
</body>
</html>
```

Canvas - Gradients
Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors. In canvas There are two different types of gradients:

1. LinearGradient
2. RadialGradient

addColorStop():
This method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.

1. createLinearGradient()
Using this method we can create a linear gradient on the canvas.

Syntax:
createLinearGradient(x,y,x1,y1)

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
</canvas>
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
// Create gradient
```

```
var grd=ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"green");
grd.addColorStop(1,"white");
// Fill with gradient
ctx.fillStyle=grd;
ctx.fillRect(10,10,150,80);
</script>
</body>
</html>
```

2. createRadialGradient(x,y,r,x1,y1,r1) - Creates a radial/circular gradient

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:2px solid #e4e4e4;">
</canvas>
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
var grd=ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"green");
grd.addColorStop(1,"white");
ctx.fillStyle=grd;
ctx.fillRect(10,10,150,80);
</script>
</body>
</html>
```

Canvas - Images

To draw an image on a canvas, we will use the following method:

1.drawImage(image,x,y)

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
```

```
<head>
<meta charset='utf-8'>
<title>
Canvas Element
</title>
</head>
<body>
<p>Image to use:</p>
<img id="scream" src="html5.png" alt="The Scream" width="220" height="277" />
<p>Canvas:</p>
<canvas id="myCanvas" width="200" height="250" style="border:1px solid #d3d3d3;">
</canvas>
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
var img=document.getElementById("scream");
ctx.drawImage(img,10,10);
</script>
</body>
</html>
```

HTML5 SVG

SVG Introduction

SVG is short for Scalable Vector Graphics. It is a graphic format in which the shapes are specified in XML. SVG is a language for describing two-dimensional vector graphics in XML. Today most web browser can display SVG just like they can display PNG, GIF, and JPG. SVG became a W3C Recommendation 14. January 2003.

1. JPEG (Joint Photographic Experts Group)
2. PNG (Portable Network Graphics)
3. GIF (Graphic Interchange Format)

SVG Versions:

SVG 1.0 became a W3C Recommendation on 4 September 2001.
SVG 1.1 became a W3C Recommendation on 14 January 2003.
SVG 1.1 (Second Edition) became a W3C Recommendation on 16 August 2011.

SVG Features?

1. W3C Standards
2. Image scaling
3. Smaller file size
4. SVG tools are already available

5. SVG images can be searched, indexed, scripted, and compressed
6. SVG images can be printed with high quality at any resolution
7. Compatibility
8. Accessibility
9. Search Engine Optimization

SVG Syntax:

```
<svg xmlns="http://www.w3.org/2000/svg">
-----------
-----------
-----------
-----------
</svg>
```

SVG Shapes

SVG has some predefined shape elements that can be used by developers:

Shape	**Keyword**
1 Rectangle	<rect>
2 Circle	<circle>
3 Ellipse	<ellipse>
4 Line	<line>
5 Polyline	<polyline>
6 Polygon	<polygon>
7 Path	<path>

Example:

HTML5 - SVG Rectangle

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<svg xmlns="http://www.w3.org/2000/svg">
<rect width="300" height="100" style="fill:rgb(0,255,0);stroke-width:1;stroke:rgb(0,0,0)" />
</svg>
  </body>
</html>
```

Code explanation:

1. The width and height attributes of the rect element define the height and the width of the rectangle
2. The style attribute is used to define CSS properties
3. The CSS fill property defines the fill color of the rectangle
4. The CSS stroke-width property defines the width of the border of the rectangle
5. The CSS stroke property defines the color of the border of the rectangle

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<svg xmlns="http://www.w3.org/2000/svg">
<rect x="50" y="20" rx="20" ry="20" width="150" height="100"
style="fill:pink;stroke:green;stroke-width:5;opacity:0.9" />
</svg>
</body>
</html>
```

Note: The rx and the ry attributes rounds the corners of the rectangle
.

SVG Circle:

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<svg xmlns="http://www.w3.org/2000/svg">
<circle cx="100" cy="50" r="50" stroke="red" fill="blue" />
</svg>
 </body>
</html>
```

1. The cx and cy attributes define the x and y coordinates of the center of the circle.

2. If cx and cy are omitted, the circle's center is set to (0, 0)
3. The r attribute defines the radius of the circle

SVG <ellipse>
The <ellipse> element is used to create an ellipse.An ellipse is closely related to a circle. The difference is that an ellipse has an x and a y radius that differs from each other, while a circle has equal x and y radius:

Example:
```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<svg xmlns="http://www.w3.org/2000/svg" >
<ellipse  cx="180"  cy="80"  rx="100"  ry="50"  style="fill:purple;stroke:purple;stroke-width:2"
/>
</svg>
</body>
</html>
```

Note:
The cx attribute defines the x coordinate of the center of the ellipse
The cy attribute defines the y coordinate of the center of the ellipse
The rx attribute defines the horizontal radius
The ry attribute defines the vertical radius

Example:
```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<svg xmlns="http://www.w3.org/2000/svg">
  <ellipse cx="240" cy="100" rx="220" ry="30" style="fill:red" />
  <ellipse cx="220" cy="70" rx="190" ry="20" style="fill:green" />
  <ellipse cx="210" cy="45" rx="170" ry="15" style="fill:blue" />
```

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

SVG <line>

The <line> element is used to create a line:

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<body>
<svg xmlns="http://www.w3.org/2000/svg">
  <line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);stroke-width:2" />
</svg>
</body>
</html>
```

Note:

The x1 attribute defines the start of the line on the x-axis
The y1 attribute defines the start of the line on the y-axis
The x2 attribute defines the end of the line on the x-axis
The y2 attribute defines the end of the line on the y-axis

SVG <polygon>

The <polygon> element is used to create a graphic that contains at least three sides. Polygons are made of straight lines, and the shape is "closed" (all the lines connect up). Polygon comes from Greek. "Poly" means "many" and "gon" means "angle".

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<body>
```

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="200,10 250,190 160,210" style="fill:red;stroke:purple;stroke-width:1" />
</svg>
</body>
</html>
```

Note: The points attribute defines the x and y coordinates for each corner of the polygon.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <polygon points="100,10 40,180 190,60 10,60 160,180" style="fill:yellow;stroke:lime;stroke-
width:5;fill-rule:nonzero;"/>
</svg>
</body>
</html>
```

SVG <polyline>

The <polyline> element is used to create any shape that consists of only straight lines.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<body>
 <svg xmlns="http://www.w3.org/2000/svg">
 <polyline points="20,20 40,25 60,40 80,120 120,140 200,180"
style="fill:none;stroke:black;stroke-width:3" />
</svg>
</body>
```

```
</html>
```

SVG <path>

The <path> element is used to define a path. The following commands are available for path data:

M = moveto
L = lineto
H = horizontal lineto
V = vertical lineto
C = curveto
S = smooth curveto
Q = quadratic Bézier curve
T = smooth quadratic Bézier curveto
A = elliptical Arc
Z = closepath

Note: All of the commands above can also be expressed with lower letters. Capital letters means absolutely positioned, lower cases means relatively positioned.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<body>
<svg xmlns="http://www.w3.org/2000/svg">
  <path d="M150 0 L75 200 L225 200 Z" />
</svg>
</body>
</html>
```

SVG Text - <text>

The <text> element is used to define a text

Example:

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

```
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<body>
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="0" y="15" fill="red">I like SVG</text>
</svg>
</body>
</html>
```

Example:
In SVG a transform refers to the geometric kind; a translation (movement), rotation, scale

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<body>
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="0" y="15" fill="blue" transform="rotate(30 20,40)">I like SVG</text>
</svg>
</body>
</html>
```

SVG Stroke Properties
SVG offers a wide range of stroke properties. They are the following:
stroke
stroke-width
stroke-linecap
stroke-dasharray

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
```

```
</head>
<body>
<h1>My SVG</h1>
<body>
<svg xmlns="http://www.w3.org/2000/svg">
  <g fill="none">
    <path stroke="red" d="M5 20 l215 0" />
    <path stroke="black" d="M5 40 l215 0" />
    <path stroke="blue" d="M5 60 l215 0" />
  </g>
</svg>
</body>
</html>
```

Example:
SVG stroke-width Property:
The stroke-width property defines the thickness of a line, text or outline of an element:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<svg xmlns="http://www.w3.org/2000/svg">
  <g fill="none" stroke="blue">
    <path stroke-width="4" d="M5 20 l215 0" />
    <path stroke-width="6" d="M5 40 l215 0" />
    <path stroke-width="8" d="M5 60 l215 0" />
  </g>
</svg>
 </body>
</html>
```

Example:
SVG stroke-linecap Property
The stroke-linecap property defines different types of endings to an open path:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
```

```
</head>
<body>
<h1>My SVG</h1>
<svg xmlns="http://www.w3.org/2000/svg">
  <g fill="none" stroke="blue" stroke-width="6">
    <path stroke-linecap="butt" d="M5 20 l215 0" />
    <path stroke-linecap="round" d="M5 40 l215 0" />
    <path stroke-linecap="square" d="M5 60 l215 0" />
  </g>
</svg>
 </body>
</html>
```

SVG Gradients
A gradient is a smooth transition from one color to another. In addition, several color transitions can be applied to the same element.

There are two main types of gradients in SVG:
Linear
Radial

SVG Linear Gradient - <linearGradient>
The <linearGradient> element is used to define a linear gradient.
The <linearGradient> element must be nested within a <defs> tag. The <defs> tag is short for definitions and contains definition of special elements (such as gradients).

Linear gradients can be defined as horizontal, vertical or angular gradients:

Horizontal gradients are created when y1 and y2 are equal and x1 and x2 differ
Vertical gradients are created when x1 and x2 are equal and y1 and y2 differ
Angular gradients are created when x1 and x2 differ and y1 and y2 differ

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <defs>
```

```
   <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
  <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
  <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
   </linearGradient>
  </defs>
  <ellipse cx="200" cy="70" rx="85" ry="55" fill="url(#grad1)" />
</svg>
 </body>
</html>
```

SVG Radial Gradient - <radialGradient>
The <radialGradient> element is used to define a radial gradient.
The <radialGradient> element must be nested within a <defs> tag. The <defs> tag is short for definitions and contains definition of special elements (such as gradients).

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <defs>
   <radialGradient id="grad1" cx="50%" cy="50%" r="50%" fx="50%" fy="50%">
<stop offset="0%" style="stop-color:rgb(255,255,255);stop-opacity:0" />
 <stop offset="100%" style="stop-color:rgb(0,0,255);stop-opacity:1" />
   </radialGradient>
  </defs>
  <ellipse cx="200" cy="70" rx="85" ry="55" fill="url(#grad1)" />
</svg>
 </body>
</html>
```

Comparison of Canvas and SVG
Canvas
1.Resolution dependent
2.Support for event handlers
3.Poor text rendering capabilities
4.You can save the resulting image as .png or .jpg
5. Suited for game applications

SVG
1. Resolution independent
2. No Support for event handlers
3. Best suited for applications with large rendering areas (GoogleMaps)
4. Slow rendering if complex
5. Not suited for game applications

SVG Drop Shadows
<defs> and <filter>

All SVG filters are defined within a <defs> element. The <defs> element is short for definitions and contains definition of special elements (such as filters).

The <filter> element is used to define an SVG filter. The <filter> element has a required id attribute which identifies the filter. The graphic then points to the filter to use.

Example:
```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h1>My SVG</h1>
<svg xmlns="http://www.w3.org/2000/svg">
  <defs>
    <filter id="f1" x="0" y="0" width="200%" height="200%">
      <feOffset result="offOut" in="SourceGraphic" dx="20" dy="20" />
      <feBlend in="SourceGraphic" in2="offOut" mode="normal" />
    </filter>
  </defs>
  <rect width="90" height="90" stroke="green" stroke-width="3"
  fill="yellow" filter="url(#f1)" />
</svg>
 </body>
</html>
```

HTML5 Microdata
Microdata is a standardized way to provide additional semantics in your web pages. Microdata lets you define your own customized elements and start embedding custom

properties in your web pages. At a high level, microdata consists of a group of name-value pairs.

Microdata defines five HTML attributes that can be applied to any HTML5 tag.
1 Itemscope - Indicates the element is a microdata element and its child elements are part of its microdata format.
2 Itemtype - Defines the vocabulary to be used by the microdata format.
3 Itemid - The unique identifier of the item, if defined by the microdata vocabulary.
4 Itemprop - An individual data element.
5 Itemref - Allows a microdata element to reference another element on the page to define it by either HTML id or by itemid.

Example:
```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
MicroData
</title>
</head>
<body>
<div itemscope>
 <img itemprop="image" src="html5.png"  alt="logohtml5">
</div>
</body>
</html>
```

Properties can also have values that are dates, times, or dates and times. This is achieved using the time element and its datetime attribute.

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
MicroData
</title>
</head>
<body>
<div itemscope>
My birthday is:
<time itemprop="birthday" datetime="1971-05-08">
```

```
 Aug 5th 1971
</time>
</div>
</body>
</html>
```

Properties can also themselves be groups of name-value pairs, by putting the itemscope attribute on the element that declares the property.

Defining Microdata Vocabulary:
To define microdata vocabulary you need a namespace URL which points to a working web page. For example http://data-vocabulary.org/Person can be used as the namespace for a personal microdata vocabulary with the following named properties:

name - Person name as a simple string
Photo - A URL to a picture of the person.
URL - A website belonging to the person.

Using about properties a person microdata could be as follows:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
MicroData
</title>
</head>
<body>
<section itemscope itemtype="http://data-vocabulary.org/Person">
<h1 itemprop="name">KSNareshITu</h1>
<p>
<img itemprop="photo" src="smiley6.jpg">
</p>
<a itemprop="url" href="http://www.nareshit.com/">My WebSite</a>
</section>
</body>
</html>
```

Microdata is a WHATWG HTML specification used to nest metadata within existing content on web pages. Search engines, web crawlers, and browsers can extract and process Microdata from a web page and use it to provide a richer browsing experience for users.

Search engines benefit greatly from direct access to this structured data because it allows search engines to understand the information on web pages and provide more relevant results to users. Microdata uses a supporting vocabulary to describe an item and name-value pairs to assign values to its properties. Microdata is an attempt to provide a simpler way of annotating HTML elements with machine-readable tags than the similar approaches of using RDFa and Microformats.

What is MathML?

Mathematical Markup Language (MathML) is an application of XML for describing mathematical notations and capturing both its structure and content. It aims at integrating mathematical formulae into World Wide Web pages and other documents. It is a recommendation of the W3C math working group.

MathML Versions:

MathML 1.0	==>April 1998
MathML 2.0	==> October 2003
MathML 3.0	==> 21st Oct-2010

Namespace:
MathML was originally designed before the finalization of XML namespaces. the elements should be in the namespace with namespace URI

Syntax:
```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  ... ... ...
  ... ... ...
</math>
```

The HTML syntax of HTML5 allows for MathML elements to be used inside a document using <math>...</math> tags.

Syntax:
```
<math>...</math>
```

Features of MathML:
- ✓ Linking
- ✓ Directionality
- ✓ Line breaking
- ✓ Including images
- ✓ Elementary math layouts
- ✓ MathML is part of HTML5
- ✓ All the browser makers are behind HTML5

- ✓ Internet Explorer has good MathML support for years
- ✓ HTML5 with MathML gives us math in all browsers and devices!

Presentation of MathML:
Presentation MathML focuses on the display of an equation, and has about 30 elements. The elements' names all begin with m. Token elements generally only contain characters (not other elements). They include:

1 <mi>x</mi> – identifiers;
2 <mo>+</mo> – operators;
3 <mn>2</mn> – numbers.
4 <mtext>non zero</mtext> – text.
5 <mrow> – a horizontal row of items;
6 <msup>, <munderover> superscripts, limits over...
7 <mfrac> – fractions;
8 <msqrt> and <mroot> – roots;

<msup> Tag:
The MathML <msup> element is used to attach a superscript to an expression.

Syntax:
<msup> base superscript </msup>.

Example:
```
<body>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<msup>
<mi>X</mi>
<mn>2</mn>
</msup>
</math>
</body>
```

<msub>
The MathML <msub> element is used to attach a subscript to an expression.

Syntax:
<msub> base subscript </msub>.

Example:
```
<body>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<msub>
```

```
<mi>X</mi>
<mn>2</mn>
</msub>
</math>
</body>
```

<msubsup>:
The MathML <msubsup> element is used to attach both a subscript and a superscript, together, to an expression.

Syntax:
<msubsup>...........................</msubsup>

Example:

```
<body>
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <msubsup>
   <mo> &#x222B;<!--Integral --> </mo>
   <mn> 0 </mn>
   <mn> 1 </mn>
  </msubsup>
 </math>
</body>
```

munder
The MathML <munder> element is used to attach an accent or a limit under an expression.

Syntax: <munder>..........................</munder>

Examples:

```
<math>
 <munder accentunder="true">
  <mrow>
   <mi> x </mi>
   <mo> + </mo>
   <mi> y </mi>
   <mo> + </mo>
   <mi> z </mi>
  </mrow>
  <mo> &#x23DF; <!--BOTTOM CURLY BRACKET--> </mo>
</munder>
 </math>
```

munderover
The MathML <munderover> element is used to attach accents or limits both under and over an expression.

syntax: <munderover> base underscript overscript </munderover>

Examples:
```
<math>
  <munderover>
   <mo> &#x222B; <!--INTEGRAL--> </mo>
   <mn> 0 </mn>
   <mi> &#x221E; <!--INFINITY--> </mi>
 </munderover>
 </math>
```

mpadded
The MathML <mpadded> element is used to add extra padding and to set the general adjustment of position and size of enclosed contents

Examples:
```
<math>
  <mpadded height="+150px" width="100px" lspace="2height">
   <mi> x </mi>
   <mo> + </mo>
   <mi> y </mi>
 </mpadded>
 </math>
```

mroot
The MathML <mroot> element is used to display roots with an explicit index. Two arguments are accepted.

Syntax:
<mroot>......................................</mroot>.

Example:
```
<body>
<math xmlns="http://www.w3.org/1998/Math/MathML">
 <mroot>
   <mi>x</mi>
   <mn>3</mn>
 </mroot>
</math>
```

```
</body>
```

msqrt
The MathML <msqrt> element is used to display square roots

Syntax: <msqrt> base </msqrt>.

Example:

```
<math>
  <msqrt>
    <mi>x</mi>
  </msqrt>
 </math>
```

<menclose>:
The MathML <menclose> element renders its content inside an enclosing notation specified by the notation attribute.

Syntax:
<menclose> -------------------</menclose>

Notation Attribute supports the Following list of values/ parameters....
1 box
2 roundedbox
3 circle
4 left
5 right
6 top
7 bottom
8 updiagonalstrike
9 downdiagonalstrike
10 verticalstrike
11 horizontalstrike
etc...

Example:

```
<body>
<math xmlns="http://www.w3.org/1998/Math/MathML">
   <menclose notation="circle box">
    <mi> x </mi>
    <mo> + </mo>
    <mi> y </mi>
  </menclose>
```

```
 </math>
</body>
```

<mfrac>
The MathML <mfrac> element is used to display fractions.

Syntax:
<mfrac>numerator denominator</mfrac>.

Example:

```
<body>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<mfrac bevelled="true">
    <mfrac>
      <mi> a </mi>
      <mi> b </mi>
    </mfrac>
    <mfrac>
      <mi> c </mi>
      <mi> d </mi>
    </mfrac>
  </mfrac>
</math>
</body>
```

MathML Examples:

```
 <!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
  <title>Pythagorean theorem</title>
  </head>
  <body>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <mrow>
        <msup><mi>a</mi><mn>2</mn></msup>
        <mo>+</mo>
        <msup><mi>b</mi><mn>2</mn></msup>
        <mo>=</mo>
        <msup><mi>c</mi><mn>2</mn></msup>
      </mrow>
    </math>
  </body>
</html>
```

Example:

```
 <!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
  <title>Pythagorean theorem</title>
  </head>
  <body>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<mrow>
 <mi>a</mi>
  <msup><mi>x</mi><mn>2</mn></msup>
 <mo>+</mo><mi>b</mi>
  <mi>x</mi><mo>+</mo><mi>c</mi>
</mrow>
</math>
</body>
</html>
```

Using MathML Characters:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
  <title>MathML Examples</title>
  </head>
  <body>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <mrow>
        <mrow>
          <msup>
            <mi>x</mi>
            <mn>2</mn>
          </msup>
          <mo>+</mo>
          <mrow>
            <mn>4</mn>
            <mi>x</mi>
          </mrow>
          <mo>+</mo>
```

```
            <mn>4</mn>
         </mrow>
            <mo>=</mo>
            <mn>0</mn>
        </mrow>
   </math>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
  <title>MathML Examples</title>
  </head>
  <body>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
       <mrow>
         <mi>A</mi>
         <mo>=</mo>
         <mfenced open="[" close="]">
           <mtable>
             <mtr>
               <mtd><mi>x</mi></mtd>
               <mtd><mi>y</mi></mtd>
             </mtr>
             <mtr>
               <mtd><mi>z</mi></mtd>
               <mtd><mi>w</mi></mtd>
             </mtr>
           </mtable>
         </mfenced>
      </mrow>
   </math>
</body>
</html>
```

Example:

```
<math display='block'>
      <mrow>
      <mfrac>
```

```
        <mrow>
          <mo>&#x2212;</mo>
          <mi>b</mi>
          <mo>&#x00B1;</mo>
          <msqrt>
            <mrow>
              <msup>
                <mi>b</mi>
                <mn>2</mn>
              </msup>
              <mo>&#x2212;</mo>
              <mn>4</mn>
              <mi>a</mi>
              <mi>c</mi>
            </mrow>
          </msqrt>
        </mrow>
        <mrow>
          <mn>2</mn>
          <mi>a</mi>
        </mrow>
      </mfrac>
    </mrow>
  </math>
</body>
```

Example:

```
<body>
    <math>
        <mrow>
  <munderover>
    <mo>&#8747;</mo>
    <mn>-1</mn>
    <mn>+1</mn>
  </munderover>
  <mfrac>
    <mrow>
      <mi>d</mi>
      <mi>x</mi>
    </mrow>
    <mi>x</mi>
  </mfrac>
</mrow>
```

```
  </math>
</body>
```

Frequently used acronyms

CSS:	Cascading Stylesheets
HTML:	Hypertext Markup Language
ISO:	International Organization for Standardization
MIME:	Multipurpose Internet Mail Extension
ODF:	OpenDocument Format
OMML:	Office math markup language
OOXML:	Office Open XML
SGML:	Standard Generalized Markup Language
UI:	User interface
URI:	Uniform Resource Identifier
W3C:	World Wide Web Consortium
XHTML:	Extensible Hypertext Markup Language
XML:	Extensible Markup Language
XSL:	Extensible Stylesheet Language
XSL-FO:	Extensible Stylesheet Language Formatting Objects
XSLT:	Extensible Stylesheet Language Transformations

HTML5 Geolocation (Deeper Integration with OS)
HTML5 Geolocation API allows users to share their location with web applications by capturing the approximate longitude and latitude coordinates of the user with their permission.

A geographic coordinate system is a coordinate system that enables every location on the Earth to be specified by a set of numbers or letters. The coordinates are often chosen such that one of the numbers represents vertical position, and two or three of the numbers represent horizontal position. A common choice of coordinates is latitude, longitude .

In geography, latitude is a geographic coordinate that specifies the north-south position of a point on the Earth's surface. Lines of constant latitude, or parallels, run east–west as circles parallel to the equator.

In geography, longitude is a geographic coordinate that specifies the east-west position of a point on the Earth's surface. It is an angular measurement, usually expressed in degrees.

Google Maps - Basic Map Types
The following map types are supported in Google Maps API:

1. ROADMAP (normal, default 2D map)
2. SATELLITE (photographic map)

3. HYBRID (photographic map + roads and city names)
4. TERRAIN (map with mountains, rivers, etc.)

The map type is specified either within the Map properties object, with the mapTypeId property:

Google Maps - The Default Controls
When showing a standard Google map, it comes with the default control set:

1. Zoom - displays a slider or "+/-" buttons to control the zoom level of the map
2. Pan - displays a pan control for panning the map
3. MapType - lets the user toggle between map types (roadmap and satellite)
4. Street View - displays a Pegman icon which can be dragged to the map to enable Street View

Google Maps - More Controls
1. Scale
2. Rotate
3. Overview Map
4. Marker
5. Polyline
6. Polygon
7. Info Windows
8. Custom overlays
9. Info Icons
10. MapPin

Google Maps - More Controls
In addition to the default controls, Google Maps also has:
1. Scale - displays a map scale element
2. Rotate - displays a small circular icon which allows you to rotate maps
3. Overview Map - displays a thumbnail overview map reflecting the current map viewport within a wider area

Google Maps - Overlays
Overlays are objects on the map that are bound to latitude/longitude coordinates.
Google Maps has several types of overlays:
Marker - Single locations on a map. Markers can also display custom icon images
Polyline - Series of straight lines on a map
Polygon - Series of straight lines on a map, and the shape is "closed"
Circle and Rectangle
Info Windows - Displays content within a popup balloon on top of a map
Custom overlays

Geolocation API:

```
<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=false
   &amp;key=ABQIAAAA8JXb0YDVa40tOLnM95w50BSeC_rwpfX9fQb-
nbMGMDH8BB4BVRTjxWS14T5WLZf7TpXaaAtk_SIb-Q"
   type="text/javascript">
  </script>
```

Note:
1. Internet Explorer 9, Firefox, Chrome, Safari and Opera support Geolocation.
2. Geolocation is much more accurate for devices with GPS, like iPhone

Geolocation Methods:
The geolocation object provides the following methods:

Method	Description
getCurrentPosition()	It retrieves the current geographic location of the user.
watchPosition()	It retrieves periodic updates about the current geographic location of the device.
clearWatch()	This method cancels an ongoing watch Position call.

Example: Check for Browser compatibility:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
Geolocation
</title>
<script type="text/javascript">
if(navigator.geolocation)
{
alert("Great..Geolocation API is supported.");
}
else
{
alert("OOPs Geolocation API is not supported in your browser Update Try...");
}
</script>
</head>
```

Get the user’s current location:

The current location of the user can be obtained using the getCurrentPosition function of the navigator.geolocation object. This function accepts three parameters – Success callback function, Error callback function and position options. If the location data is fetched successfully, the success callback function will be invoked with the obtained position object as its input parameter. Otherwise, the error callback function will be invoked with the error object as its input parameter.

Example:

```
<script>
function MyPosition()
{
navigator.geolocation.getCurrentPosition(function(position)
{
do_something(position.coords.latitude, position.coords.longitude);
});
}
</script>
```

Example:

```
<body>
<p id="demo">Click the button to get your coordinates:</p>
<button onclick="getLocation()">GetCurrentPosition</button>
<script>
var x = document.getElementById("demo");
function getLocation()
{
if (navigator.geolocation)
{
navigator.geolocation.getCurrentPosition(showPosition);
}
else
{
x.innerHTML = "Geolocation is not supported by this browser.";
   }
}
function showPosition(position)
{
x.innerHTML = "Latitude: " + position.coords.latitude +
"<br>Longitude: " + position.coords.longitude;
}
</script>
</body>
```

Error callback function (Handling Errors and Rejections)
This is an optional callback function that takes a 'Position Error' object as its input parameter. This function is invoked under any one of the following circumstances
 Unknown Error occurred
 Request timed out
 User has denied to share the location information
 Location information itself is unavailable

Example:
```
function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      alert("User denied the request for Geolocation.");
      break;
    case error.POSITION_UNAVAILABLE:
      alert("Location information is unavailable.");
      break;
    case error.TIMEOUT:
      alert("The request to get user location timed out.");
      break;
    case error.UNKNOWN_ERROR:
      alert("An unknown error occurred.");
      break;
  }
}
```

Show My Location on Google Maps:
First of all, we should convert the latitude and longitude coordinates of the position object obtained using the Geolocation API into a Google maps latLng object.

Example:
```
<head>
<script>
var x = document.getElementById("demo");
function getLocation() {
if (navigator.geolocation)
{
navigator.geolocation.getCurrentPosition(showPosition,showError);
}
else
{
x.innerHTML = "Geolocation is not supported by this browser.";
}
}
function showPosition(position)
{
var latlon = position.coords.latitude+","+position.coords.longitude;
 var img_url = "http://maps.googleapis.com/maps/api/staticmap?center="
   +latlon+"&zoom=14&size=400x300&sensor=false";
   document.getElementById("mapholder").innerHTML = "<img src='"+img_url+"'>";
}
function showError(error)
{
switch(error.code) {
case error.PERMISSION_DENIED:
x.innerHTML = "User denied the request for Geolocation."
break;
case error.POSITION_UNAVAILABLE:
x.innerHTML = "Location information is unavailable."
break;
case error.TIMEOUT:
x.innerHTML = "The request to get user location timed out."
break;
case error.UNKNOWN_ERROR:
x.innerHTML = "An unknown error occurred."
break;
}
}
</script>
</head>
<body>
<p id="demo">Click the button to get your position:</p>
<button onclick="getLocation()">Display_Map</button>
<div id="mapholder"></div>
</body>
```

Example (Geolocation Live Example)

```
<!DOCTYPE html>
<html>
<head>
<title>Sample Map</title>
<style>
#mapdiv {
   margin: 0;
   padding: 0;
   width: 500px;
   height: 500px;
}
</style>
<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=true"></script>
<script>
   var watchId = null;
   function geoloc() {
   if (navigator.geolocation) {
      var optn = {
            enableHighAccuracy : true,
            timeout : Infinity,
            maximumAge : 0
      };
   watchId = navigator.geolocation.watchPosition(showPosition, showError, optn);
   } else {
         alert('Geolocation is not supported in your browser');
   }
   }
 function showPosition(position) {
      var        googlePos        =        new        google.maps.LatLng(position.coords.latitude,
position.coords.longitude);
      var mapOptions = {
         zoom : 12,
         center : googlePos,
         mapTypeId : google.maps.MapTypeId.ROADMAP
      };
      var mapObj = document.getElementById('mapdiv');
      var googleMap = new google.maps.Map(mapObj, mapOptions);
      var markerOpt = {
         map : googleMap,
         position : googlePos,
         title : 'Hi , I am here',
         animation : google.maps.Animation.DROP
      };
      var googleMarker = new google.maps.Marker(markerOpt);
      var geocoder = new google.maps.Geocoder();
      geocoder.geocode({
         'latLng' : googlePos
         }, function(results, status) {
            if (status == google.maps.GeocoderStatus.OK) {
            if (results[1]) {
               var popOpts = {
```

```
            content : results[1].formatted_address,
            position : googlePos
          };
        var popup = new google.maps.InfoWindow(popOpts);
        google.maps.event.addListener(googleMarker, 'click', function() {
        popup.open(googleMap);
    });
        } else {
          alert('No results found');
        }
        } else {
          alert('Geocoder failed due to: ' + status);
        }
    });
    }

    function stopWatch() {
        if (watchId) {
          navigator.geolocation.clearWatch(watchId);
          watchId = null;

        }
    }

function showError(error) {
var err = document.getElementById('mapdiv');
switch(error.code) {
case error.PERMISSION_DENIED:
err.innerHTML = "User denied the request for Geolocation."
break;
case error.POSITION_UNAVAILABLE:
err.innerHTML = "Location information is unavailable."
break;
case error.TIMEOUT:
err.innerHTML = "The request to get user location timed out."
break;
case error.UNKNOWN_ERROR:
err.innerHTML = "An unknown error occurred."
break;
}
}
</script>
```

```
  </head>
  <body onload="geoloc()">
    <p id = 'mapdiv'></p>
    <button onclick="stopWatch()">
      Stop
    </button>
  </body>
</html>
```

HTML5 Drag and Drop (Deeper Integration with OS)

Drag and Drop (DnD) is powerful User Interface concept which makes it easy to copy, reorder and deletion of items with the help of mouse clicks. This allows the user to click and hold the mouse button down over an element, drag it to another location, and release the mouse button to drop the element there.

The Drag and Drop Event is the most fabulous properties of HTML5. With the help of Drag and Drop features We can move an object from one place to another place. The all major browsers support. All the process is done with the help of Java Script. The Drag and Drop Event may be a little bit complex.

Creating draggable content

Making an object draggable is simple. Set the draggable=true attribute on the element you want to make moveable. Just about anything can be drag-enabled, including images, links, files, or other DOM nodes.

Example:

```
<div id="columns">
  <div class="column" draggable="true"><header>A</header></div>
  <div class="column" draggable="true"><header>B</header></div>
  </div>
```

Drag and Drop Events:

Events	**Description**
dragstart	Fires when the user starts dragging of the object.
dragenter	Fires when the mouse is first moved over the target element
dragover	This event is fired as the mouse is moved over an element
dragleave	This event is fired when the mouse leaves an element
drag	Fires every time the mouse is moved while the object is being dragged
drop	The drop event is fired on the element where the drop was occured.
dragend	Fires when the user releases the mouse button while dragging an object.

Where to Drop - ondragover
The ondragover event specifies where the dragged data can be dropped. By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.

```
function allowDrop(ev)
{
   ev.preventDefault();
}
```

What to Drag - ondragstart and setData()
The dataTransfer.setData() method sets the data type and the value of the dragged data:

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

In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

Do the Drop - ondrop
When the dragged data is dropped, a drop event occurs. the ondrop attribute calls a function, drop(event):

```
function drop(ev)
{
   ev.preventDefault();
   var data = ev.dataTransfer.getData("text");
   ev.target.appendChild(document.getElementById(data));
}
```

Example:

```
<head>
<style>
#div1
{
width:350px;height:70px;
padding:10px;
border:1px solid #ff00ff;
}
</style>
<script type='text/javascript'>
function allowDrop(ev)
{
   ev.preventDefault();
}

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

function drop(ev)
{
   ev.preventDefault();
   var data = ev.dataTransfer.getData("text/html");
   ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<br>
<img id="drag1" src="water.gif" draggable="true" ondragstart="drag(event)" width="336"
height="69">
</body>
```

Example:2

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
DragnDrop
</title>
<style type="text/css">
#div1, #div2
{
float:left;
width:150px;
height:100px;
margin:10px;
padding:10px;
border:2px solid #FF0000;
}
</style>
<script type='text/javascript'>
```

```
function allowDrop(ev)
{
ev.preventDefault();
}
function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">
 <img src="html5.png" draggable="true" ondragstart="drag(event)" id="drag1"
width="150" height="100" />
</div>
<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
</body>
</html>
```

Drag and Drop Process:
Following are the steps to be carried out to implement Drag and Drop operation:

Step 1: Making an Object Draggable:
Here are steps to be taken:
If you want to drag an element, you need to set the draggable attribute to true for that element.Set an event listener for dragstart that stores the data being dragged.The event listener dragstart will set the allowed effects (copy, move, link, or some combination).

Example:

```
<head>
<style type="text/css">
#boxA, #boxB
{
float:left;padding:10px;margin:10px; -moz-user-select:none;
}
#boxA
{
```

```
background-color: #6633FF;
width:75px;
height:75px;
}
#boxB
{
background-color: #FF6699;
width:150px;
height:150px;
}
</style>
<script type="text/javascript">
function dragStart(ev) {
  ev.dataTransfer.effectAllowed='move';
  ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
  ev.dataTransfer.setDragImage(ev.target,0,0);
  return true;
}
</script>
</head>
<body>
<center>
<h1>Drag and drop HTML5 demo</h1>
<div>Try to drag the purple box around.</div>

<div id="boxA" draggable="true"
     ondragstart="return dragStart(event)">
  <p>Drag Me</p>
</div>
<div id="boxB">Dustbin</div>
</center>
</body>
```

Step 2: Dropping the Object:
To accept a drop, the drop target has to listen to at least three events:

1 The dragenter event, which is used to determine whether or not the drop target is to accept the drop. If the drop is to be accepted, then this event has to be canceled.

2 The dragover event, which is used to determine what feedback is to be shown to the user. If the event is canceled, then the feedback (typically the cursor) is updated based on the dropEffect attribute's value.

3 Finally, the drop event, which allows the actual drop to be performed.

Example:

```
<head>
<style type="text/css">
#boxA, #boxB {
  float:left;padding:10px;margin:10px;-moz-user-select:none;
}
#boxA { background-color: #6633FF; width:75px; height:75px;  }
#boxB { background-color: #FF6699; width:150px; height:150px; }
</style>
<script type="text/javascript">
function dragStart(ev) {
  ev.dataTransfer.effectAllowed='move';
  ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
  ev.dataTransfer.setDragImage(ev.target,0,0);
  return true;
}
function dragEnter(ev) {
  event.preventDefault();
  return true;
}
function dragOver(ev) {
   return false;
}
function dragDrop(ev) {
  var src = ev.dataTransfer.getData("Text");
  ev.target.appendChild(document.getElementById(src));
  ev.stopPropagation();
  return false;
}
</script>
</head>
<body>
<center>
<h2>Drag and drop HTML5 demo</h2>
<div>Try to move the purple box into the pink box.</div>

<div id="boxA" draggable="true"
    ondragstart="return dragStart(event)">
  <p>Drag Me</p>
</div>
<div id="boxB" ondragenter="return dragEnter(event)"
```

```
    ondrop="return dragDrop(event)"
    ondragover="return dragOver(event)">Dustbin</div>
</center>
</body>
```

JS Cookies
Cookies were originally invented by Netscape to give 'memory' to web servers and browsers. Cookies are intended to help you access a site faster and more efficiently. Cookies are usually small text files, Cookies are created when you use your browser to visit a website that uses cookies to keep track of your movements within the site.

There are two types of cookies:
1. Session cookies
2. Persistent cookies.

Session cookies are created temporarily in your browser's subfolder while you are visiting a website. Once you leave the site, the session cookie is deleted.

Persistent cookie files remain in your browser's subfolder and are activated again once you visit the website that created that particular cookie.

A cookie can only hold string based name/value pairs (i.e. name=value settings). You cannot store binary data in a cookie. Cookies can be a maximum of 4kb in size each. A single server or domain can only store a total of 20 cookies per user browser.

How It Works ?
Your server sends some data to the visitor's browser in the form of a cookie. The browser may accept the cookie. If it does, it is stored as a plain text record on the visitor's hard drive. Now, when the visitor arrives at another page on your site, the browser sends the same cookie to the server for retrieval. Once retrieved, your server knows/remembers what was stored earlier.

Advantages
1. Cookies do not require any server resources since they are stored on the client.
2. Cookies are easy to implement.
3. You can configure cookies to expire when the browser session ends (session cookies) or they can exist for a specified length of time on the client computer (persistent cookies).

Disadvantages
1. User can delete a cookies.
2. Cookies can be disabled on user browsers
3. No security for sensitive data
5) Most browsers support cookies of up to 4096 bytes(4kbytes)

6)Most browsers allow only 20 cookies per site; if you try to store more, the oldest cookies are discarded.
7)Browser supports 300 cookies towards different websites.
8)Cookies are browser specific (ie, one browser type[IE] stored cookies will not be used by another browser type[firefox]).

Cookies are a plain text data record of 5 variable-length fields:

Expires : The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.

Domain : The domain name of your site.

Path : The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.

Secure : If this field contains the word "secure" then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.

Name=Value : Cookies are set and retrieved in the form of key and value pairs.

The Structure of a Cookie
Cookies are created using the cookie property of the Document object. The format of a cookie is as follows:
name=value; expires=expirationDateGMT; path=URLpath; domain=siteDomain

Web SQL Database is a web page API for storing data in databases that can be queried using a variant of SQL. The API is supported by Google Chrome, Opera, Safari and the Android Browser.

An IndexedDB is basically a persistent data store in the browser—a database on the client side. Like regular relational databases, it maintains indexes over the records it stores and developers use the IndexedDB JavaScript API to locate records by key or by looking up an index.

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private.

Client-Side Storage (Expect the unexpected)

It has the following alias Names:
1. Web storage
2. Offline Storage
3. Local Storage etc...

HTML5 web storage, a better local storage than cookies. Local Storage is intended to be used for storing and retrieving data in html pages from the same domain. With HTML5, web pages can store data locally within the user's browser. Earlier, this was done with cookies.

Web Storage is more secure and faster.It is also possible to store large amounts of data, without affecting the website's performance. The data is stored in key/value pairs, and a web page can only access data stored by itself. It can store up to 10MB of data per domain. Unlike cookies, the data stored are not included with every HTTP request.

Types of Web Storage: There are two new objects for storing data on the client:
1 code.sessionStorage - stores data for one session (data is lost when the tab is closed)
2 window.localStorage - stores data with no expiration date

Session storage:
Stores data for one session. Data persisted will be cleared as soon as the user closes the browser.

Local storage:
Stores data with no expiration date. The data will be available even when the browser/ browsing tab is closed or reopened.

Note:
In both the cases, please note that the web storage data will not be available between different browsers.

Browser Support:
```
<!doctype html>
<body>
<div id="result"></div>
<script type='text/javascript'>
if (typeof(Storage) != "undefined")
{
localStorage.name="Success Browser supports";     // Store
document.getElementById("result").innerHTML = localStorage.name;  // Retrieve
}
else
{
```

```
document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Storage...";
}
</script>
</body>
```

Example:

```
<head>
<script type='text/javascript'>
var sno=100
alert("The Entered Number is: " +sno);
</script>
</head>
<body>
<a href="page2.html">Page2</a>
</body>
```

Example2

```
<head>
<script type='text/javascript'>
alert("hi")
alert(sno)
</script>
</head>
```

SessionStorage:

It is also used the store the browser's data locally, but it is for limited period (for one session), when we close the browser, it will automatically delete the stored data and we can't see the browser's stored data again. HTML5 introduces the sessionStorage attribute which would be used by the sites to add data to the session storage.

Example:

```
<head>
<script type='text/javascript'>
sessionStorage.sno=100
alert(sessionStorage.sno)
</script>
</head>
<body>
<a href="page2.html">NextPage</a>
</body>
```

Example2

```
<head>
<script type='text/javascript'>
alert("hi")
alert(sessionStorage.sno)
</script>
</head>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
session storage
</title>
</head>
<body>
  <script type="text/javascript">
    if( sessionStorage.hits )
{
sessionStorage.hits = Number(sessionStorage.hits) +1;
}
else
{
sessionStorage.hits = 1;
}
    document.write("Total Hits :" + sessionStorage.hits );
  </script>
  <p>Refresh the page to increase number of hits.</p>
  <p>Close the window and open it again and check the result.</p>
</body>
</html>
```

Example:

```
<head>
<script>
function clickCounter()
{
    if(typeof(Storage) !== "undefined") {
        if (sessionStorage.clickcount) {
            sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
        } else {
            sessionStorage.clickcount = 1;
```

```
      }
      document.getElementById("result").innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s).";
   } else {
      document.getElementById("result").innerHTML = "Sorry, your browser does not
support web storage...";
   }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
</body>
```

Local Storage:
Which is used to store the data locally. It stores the data with no expiration date, means if browser is closed then it will not delete the data and we can view the data any time and even after year.

Example:

```
<head>
<script type='text/javascript'>
localStorage.sno=100
alert(localStorage.sno)
</script>
</head>
<body>
<a href="page2.html">Gotonext</a>
</body>
```

Example2:

```
<head>
<script type='text/javascript'>
alert("hi")
alert(localStorage.sno)
</script>
</head>
```

Example3:

```
<p> You have viewed this page
  <span id="count"></span>  time(s).
</p>
```

```
<script type='text/javascript'>
 if (!localStorage.pageLoadCount)
 localStorage.pageLoadCount = 0;
 localStorage.pageLoadCount = parseInt(localStorage.pageLoadCount) + 1;
 document.getElementById('count').textContent = localStorage.pageLoadCount;
</script>
```

Example:

```
<!DOCTYPE html>
<html lang='en-US'>
<head>
<meta charset='utf-8'>
<title>
LocalStorage
</title>
<body>
 <script type="text/javascript">
  if( localStorage.hits )
   {
    localStorage.hits = Number(localStorage.hits) +1;
  }
  else
  {
    localStorage.hits = 1;
  }
  document.write("Total Hits :" + localStorage.hits );
 </script>
 <p>Refresh the page to increase number of hits.</p>
 <p>Close the window and open it again and check the result.</p>
</body>
</html>
```

Example:

```
<head>
<script>
function clickCounter()
{
  if(typeof(Storage) !== "undefined") {
    if (localStorage.clickcount) {
      localStorage.clickcount = Number(localStorage.clickcount)+1;
    } else {
      localStorage.clickcount = 1;
    }
    document.getElementById("result").innerHTML = "You have clicked the button " +
localStorage.clickcount + " time(s).";
  } else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not
support web storage...";
  }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
</body>
```

Delete Web Storage:
Storing sensitive data on local machine could be dangerous and could leave a security hole.
The Session Storage Data would be deleted by the browsers immediately after the session gets terminated.

To clear a local storage setting you would need to call localStorage.remove('key'); where 'key' is the key of the value you want to remove. If you want to clear all settings, you need to call localStorage.clear() method.

Example:

```
<body>
 <script type="text/javascript">
  localStorage.clear();
  // Reset number of hits.
  if( localStorage.hits ){
    localStorage.hits = Number(localStorage.hits) +1;
  }else{
    localStorage.hits = 1;
  }
  document.write("Total Hits :" + localStorage.hits );
 </script>
 <p>Refreshing the page would not to increase hit counter.</p>
 <p>Close the window and open it again and check the result.</p>
</body>
```

Strengths of Web Storage
1.Supported on all modern browsers, as well as on iOS and Android
2.Simple API signature.
3.Simple call flow, being a synchronous API.

4.Semantic events available to keep other tabs/windows in sync.

Weakness of Web Storage
1.Poor performance for large/complex data,
2.Poor performance when searching large/complex data, due to lack of indexing.
3.Poor performance when storing and retrieving large/complex data structures
4.Need to ensure data consistency and integrity, since data is effectively unstructured.

Example:
```
<script type='text/javascript'>
function fun1()
{
if(document.getElementById('chk1').checked==true)
{
nm=document.getElementById('t1').value
val=document.getElementById('t2').value
localStorage.setItem(nm,val)
}
}
function fun2()
{
nm=document.getElementById('t1').value
document.getElementById('t2').value=localStorage.getItem(nm)
}
</script>
Username:<input id='t1' onblur="fun2()">
<br>
Password:<input id='t2' type="password">
<br>
<input type='checkbox' id='chk1'>Remember me
<br>
<input type="button" value="Click" onclick="fun1()">
```

HTML5 Server-Sent Events
The type of events which are flowing from web browser to the web server may be called client-sent events. HTML5 Server-Sent Events allow a web page to get updates from a server.

Server-Sent Events - One Way Messaging

A server-sent event is when a web page automatically gets updates from a server. This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

Examples:
Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

Note:
Server-Sent Events are supported all major web browsers.

The EventSource Object
In the examples above we used the onmessage event to get messages. But other events are also available:

Events	**Description**
onopen	When a connection to the server is opened
onmessage	When a message is received
onerror	When an error occurs

Example:
```
<h1>Getting server updates</h1>
<div id="result"></div>
<script>
if(typeof(EventSource) !== "undefined") {
   var source = new EventSource("scores.php");
   source.onmessage = function(event) {
      document.getElementById("result").innerHTML = event.data;
   };
} else {
   document.getElementById("result").innerHTML = "Sorry, your browser does not support
server-sent events...";
}
</script>
</body>
```

Example(php)
```
<?php
header('Content-Type: text/event-stream');
$str="<h1 style=color:Red>Wickets:6</h1>
<h1 style=color:green>Runs:55</h1>
<h1 style=color:blue>Overs:14</h1>";
echo "data:$str\n\n";
?>
```

HTML5 Web Workers

JavaScript was designed to run in a single-threaded environment, meaning multiple scripts cannot run at the same time. Consider a situation where you need to handle UI events, query and process large amounts of API data, and manipulate the DOM. Javascript will hang your browser in situation where CPU utilization is high.

Example:

```
<head>
 <script>
   function bigLoop()
{
     for (var i = 0; i <= 10000000000; i += 1){
       var j = i;
     }
     alert("Completed " + j + "iterations" );
   }
 </script>
</head>
<body>
<input type="button" onclick="bigLoop();" value="BLoop" />
</body>
```

What is Web Workers?

A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

Web Workers allow for long-running scripts that are not interrupted by scripts that respond to clicks or other user interactions, and allows long tasks to be executed without yielding to keep the page responsive.

Checking for Browser Support:

Following is the syntax to detect a Web Worker feature support available in a browser:

Example: (http://modernizr.com/download)

```
<!doctype html>
<head>
 <script src="modernizr-1.5.min.js"></script>
 <script>
  if (Modernizr.webworkers)
{
```

```
alert("Congratulation!! you have web workers support." );
}
else
{
alert("Sorry!! you do not have web workers support." );
}
</script>
</head>
```

How Web Workers Work?
Web Workers are initialized with the URL of a JavaScript file, which contains the code the worker will execute. This code sets event listeners and communicates with the script that spawned it from the main page.

Syntax:
```
var worker = new Worker('name.js');
```

Once the Web Worker is spawned, communication between web worker and its parent page is done using the postMessage() method. Message passed by Web Worker is accessed using onmessage event in the main page.

Example:
```
<head>
  <script>
     var worker = new Worker('bigLoop.js');
     worker.onmessage = function (event) {
       alert("Completed " + event.data + "iterations" );
     };

     function sayHello(){
        alert("Hello sir...." );
     }
  </script>
</head>
<body>
   <input type="button" onclick="sayHello();" value="Say Hello"/>
</body>
```

Create a Web Worker File
Now, let's create our web worker in an external JavaScript. Here, we create a script that counts. The script is stored in the "jsworkers.js" file:

Example: (Save with .js Extension)

```
var i=0;
function timedCount()
{
i=i+1;
postMessage(i);
setTimeout("timedCount()",1000);
}
timedCount();
```

Example2: (save with.html)

```
<script>
w = new Worker("jsworker.js");
w.onmessage = function(event){
  document.getElementById("result").innerHTML = event.data;
  };
</script>
<div id="result"></div>
<input>
```

Terminate a Web Worker
When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated. To terminate a web worker, and free browser /computer resources, use the terminate() method

```
w.terminate();
```

A terminated Web Worker will no longer respond to messages or perform any additional computations. You cannot restart a worker; instead, you can create a new worker using the same URL.

Reuse the Web Worker
If you set the worker variable to undefined, after it has been terminated, you can reuse the code:

```
w = undefined;
```

Full Web Worker Example:

```
<body>
<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>
<script>
var w;
```

```
function startWorker() {
  if(typeof(Worker) !== "undefined") {
    if(typeof(w) == "undefined") {
    w = new Worker("jsworker.js");
    }
    w.onmessage = function(event) {
    document.getElementById("result").innerHTML = event.data;
    };
  } else {
document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Workers...";
  }
}
function stopWorker() {
  w.terminate();
  w = undefined;
}
</script>
</body>
```

HTML5 Application Cache
HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection.

Using the cache interface gives your application three advantages:
Offline browsing - users can navigate your full site when they're offline
Speed - resources come straight from disk, no trip to the network.
Resilience - if your site goes down for "maintenance" (as in, someone accidentally breaks everything), your users will get the offline experience. (ability to recover)

The Application Cache (or AppCache) allows a developer to specify which files the browser should cache and make available to offline users. Your app will load and work correctly, even if the user presses the refresh button while they're offline.

Note: Application cache is supported in all major web Browsers.

Cache Manifest Basics:
THE CACHE MANIFEST FILE
The cache manifest file is a simple text file that lists the resources the browser should cache for offline access.

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
```

...
</html>

Note:The recommended file extension for manifest files is: ".appcache"

The manifest file has three sections:
1. CACHE MANIFEST - Files listed under this header will be cached after they are downloaded for the first time
2. NETWORK - Files listed under this header require a connection to the server, and will never be cached
3. FALLBACK - Files listed under this header specifies fallback pages, if a page is inaccessible

Step1: Creating appcache file:

```
CACHE MANIFEST
# 2014-11-28:v2
main.js (anyfilename)
fish1.gif (Anyimage)
apptest.css (anyfilename)

# Use from network if available
NETWORK:
*
.php,asp,jsp, cgi (AnyOneResources)

# Fallback content
FALLBACK:
Offlinefall.html (AnyFileName-ErrorResources-Exception)
```

NOTE: # means Comment or Comment(s)

Step2: Create JS file

```
alert("HTML5 App Cache");
```

Step3: Create CSS file

```
div
{
width:100px;height:100px;
background:red;position:relative;
animation:myfirst 5s infinite;
animation-direction:alternate;
}
@keyframes myfirst
```

```
{
0%   {background:red; left:0px; top:0px;}
25%  {background:yellow; left:200px; top:0px;}
50%  {background:blue; left:200px; top:200px;}
75%  {background:green; left:0px; top:200px;}
100% {background:red; left:0px; top:0px;}
}
@-moz-keyframes myfirst /* Firefox */
{
0%   {background:red; left:0px; top:0px;}
25%  {background:yellow; left:200px; top:0px;}
50%  {background:blue; left:200px; top:200px;}
75%  {background:green; left:0px; top:200px;}
100% {background:red; left:0px; top:0px;}
}
```

Step4: Create one Image file
chrome.png;

NOTE:
The manifest file above lists three resources: a CSS file, a GIF image, and a JavaScript file. When the manifest file is loaded, the browser will download the three files from the root directory of the web site. Then, whenever the user is not connected to the internet, the resources will still be available.

Step5:
# The NETWORK section below specifies that the file "login.php" should never be cached, and will not be available offline:
NETWORK:
*
An asterisk can be used to indicate that all other resources/files require an internet connection:

(IIS, Tomcat, Apache, JigSaw etc.... from Any Web Server)
Internet Information Services (Microsoft for .NET Techs.)
Tomcat ==> Java Technologies
Apache ==> JAVA and PHP Technologies
JigSaw ==> W3C Web Server...

login.php

```
<?php
$uname=$_POST['txtuname'];
$pwd=$_POST['txtpwd'];
```

```
mysql_connect("localhost","root","");;
mysql_select_db("test");
$data=mysql_query("select * from tbl_user where uname='$uname' and pwd='$pwd'");
if(mysql_num_rows($data)==0)
echo "Invalid User";
else
echo "Valid User";
?>
```

Step6:
FALLBACK:
The FALLBACK section below specifies that "Offlinefall.html" will be served in place of all files, in case an internet connection cannot be established:
Offlinefall.html

```
<body>
<h1>Sorry File Not Existed...@@</h1>
</body>
```

Step7: Create .html file with the above resources:

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
<img src="fish1.gif">
<script type="text/javascript" src="main.js"></script>
<link rel="stylesheet" href="apptest.css">
<body>
<div></div>
</body>
</html>
```

XAMPP
X ==> AnyOS or Linux
A ==> Apache
M ==> MySQL
P ==> PERL (Practical Extration Report Language)
P ==> PHP (HyperTextPreProcessor-PersonalHomePage)

Updating the Cache
Once an application is cached, it remains cached until one of the following happens:
1 The user clears the browser's cache
2 The manifest file is modified
3 The application cache is programmatically updated

Note:
Lines starting with a "#" are comment lines, but can also serve another purpose. An application's cache is only updated when its manifest file changes. If you edit an image or change a JavaScript function, those changes will not be re-cached.

Cache states
Each application cache has a state, which indicates the current condition of the cache.

1 UNCACHED
A special value that indicates that an application cache object is not fully initialized.
2 IDLE
The application cache is not currently in the process of being updated.
3 CHECKING
The manifest is being fetched and checked for updates.
4 DOWNLOADING
Resources are being downloaded to be added to the cache, due to a changed resource manifest.
5 UPDATEREADY
There is a new version of the application cache available. There is a corresponding updateready event, which is fired instead of the cached event when a new update has been downloaded but not yet activated using the swapCache() method.
6 OBSOLETE
The application cache group is now obsolete.

Example:
```
var appCache = window.applicationCache;

switch (appCache.status) {
  case appCache.UNCACHED: // UNCACHED == 0
    return 'UNCACHED';
    break;
  case appCache.IDLE: // IDLE == 1
    return 'IDLE';
    break;
  case appCache.CHECKING: // CHECKING == 2
    return 'CHECKING';
    break;
  case appCache.DOWNLOADING: // DOWNLOADING == 3
    return 'DOWNLOADING';
    break;
  case appCache.UPDATEREADY:  // UPDATEREADY == 4
    return 'UPDATEREADY';
```

```
   break;
  case appCache.OBSOLETE: // OBSOLETE == 5
   return 'OBSOLETE';
   break;
  default:
   return 'UKNOWN CACHE STATUS';
   break;
};
```

HTML5 - WebSockets

Web Sockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers.

Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a send() method, and receive data from server to browser by an onmessage event handler.

Following is the API which creates a new WebSocket object.

Syntax:

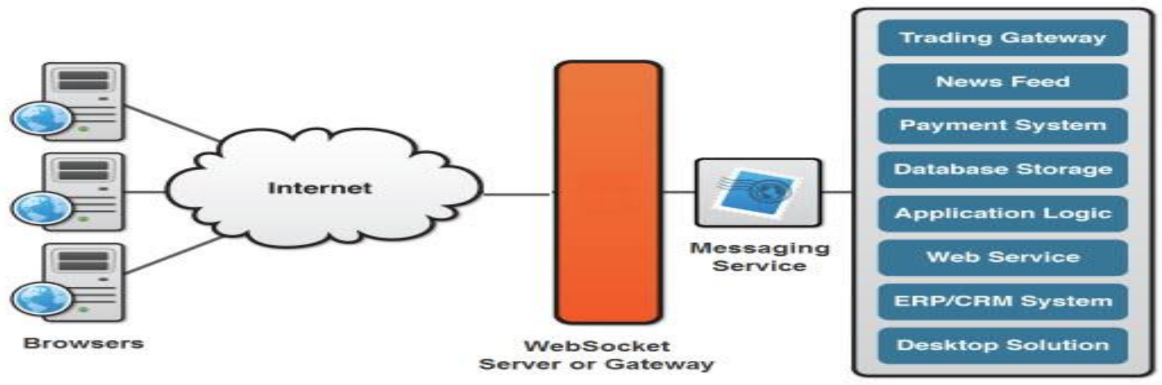
```
var Socket = new WebSocket(url, [protocal] );
```

Example:

```
var myWebSocket = new WebSocket("ws://www.websockets.org");
```

Here first argument, url, specifies the URL to which to connect. The second attribute, protocol is optional, and if present, specifies a sub-protocol that the server must support for the connection to be successful.

Websocket-architecture:-



WebSocket Attributes:
Following are the attribute of WebSocket object.

Attribute	**Description**
Socket.readyState	The readonly attribute readyState represents the state of the connection 0 indicates that the connection has not yet beenestablished. 1 indicates that the connection is established and comm. is possible. 2 indi. that the connection is going through the closing handshake 3 indi. that the connection has been closed or could not be opened
Socket.bufferedAmount	The readonly attribute bufferedAmount represents the number of bytes of UTF-8 text that have been queued using send() method.

CONNECTING (numeric value 0)
The connection has not yet been established.
OPEN (numeric value 1)
The WebSocket connection is established and communication is possible.
CLOSING (numeric value 2)
The connection is going through the closing handshake, or the close() method has been invoked.
CLOSED (numeric value 3)
The connection has been closed or could not be opened.

WebSocket Events:
Following are the events associated with WebSocket object.

Event	Event Handler	Description
open	Socket.onopen	This event occurs when socket connection is established.
Message	Socket.onmessage	This event occurs when client receives data from server.
error	Socket.onerror	This event occurs when there is any error in communication.
close	Socket.onclose	This event occurs when connection is closed

```
myWebSocket.onopen = function(evt) { alert("Connection open ..."); };
myWebSocket.onmessage = function(evt) { alert( "Received Message: "  +  evt.data); };
myWebSocket.onclose = function(evt) { alert("Connection closed."); };
```

WebSocket Methods:
Following are the methods associated with WebSocket object. Assuming we created Socket object as mentioned above:

Method	Description
Socket.send()	The send(data) method transmits data using the connection.

Socket.close() The close() method would be used to terminate any existing connection.

```
myWebSocket.send("Hello WebSockets!");
myWebSocket.close();
```

Install pywebsocket:
Before you test above client program, you need a server which supports WebSocket. Download mod_pywebsocket-x.x.x.tar.gz from pywebsocket which aims to provide a Web Socket extension for Apache HTTP Server ans install it following these steps. (https://code.google.com/p/pywebsocket/downloads/list)

1 Unzip and untar the downloaded file.
2 Go inside pywebsocket-x.x.x/src/ directory.
3 $python setup.py build
4 $sudo python setup.py install
5 Then read document by:
    $pydoc mod_pywebsocket

Start the Server
Go to the pywebsocket-x.x.x/src/mod_pywebsocket folder and run the following command:
$sudo python standalone.py -p 9998 -w ../example/

Example: (Browser Support)

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function WebSocketTest()
{
 if ("WebSocket" in window)
 {
   alert("WebSocket is supported by your Browser!");
    }
 else
 {
   alert("WebSocket NOT supported by your Browser!");
 }
}
</script>
</head>
<body>
```

```
<p>Click the button to display the result as per browser settings...!!</p>
<button onclick="WebSocketTest()">Test_Socket</button>
</body>
</html>
```

WebSocket Example:
A WebSocket is a standard bidirectional TCP socket between the client and the server. The socket starts out as a HTTP connection and then "Upgrades" to a TCP socket after a HTTP handshake. After the handshake, either side can send data.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function WebSocketTest()
{
 if ("WebSocket" in window)
 {
   alert("WebSocket is supported by your Browser!");
   // Let us open a web socket
   var ws = new WebSocket("ws://localhost:9998/echo");
   ws.onopen = function()
   {
     // Web Socket is connected, send data using send()
     ws.send("Message to send");
     alert("Message is sent...");
   };
   ws.onmessage = function (evt)
   {
     var received_msg = evt.data;
     alert("Message is received...");
   };
   ws.onclose = function()
   {
     // websocket is closed.
     alert("Connection is closed...");
   };
 }
 else
 {
   // The browser doesn't support WebSocket
   alert("WebSocket NOT supported by your Browser!");
 }
}
</script>
</head>
<body>
<div id="sse">
  <a href="javascript:WebSocketTest()">Run WebSocket</a>
</div>
</body>
</html>
```

http://datatracker.ietf.org/doc/rfc6455/?include_text=1
The protocol has two parts: a handshake and the data transfer.

```
  The handshake from the client looks as follows:
      GET /chat HTTP/1.1
      Host: server.example.com
      Upgrade: websocket
      Connection: Upgrade
      Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
      Origin: http://example.com
      Sec-WebSocket-Protocol: chat, superchat
      Sec-WebSocket-Version: 13

  The handshake from the server looks as follows:
      HTTP/1.1 101 Switching Protocols
      Upgrade: websocket
      Connection: Upgrade
      Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
      Sec-WebSocket-Protocol: chat
```

HTML5 - Web SQL Database

The Web SQL Database API isn't actually part of the HTML5 specification but it is a separate specification which introduces a set of APIs to manipulate client-side databases using SQL.

Note: Web SQL Database will work in latest version of Safari, Chrome and Opera.

The Core Methods:

There are following three core methods:

1. openDatabase: This method creates the database object either using existing database or creating new one.
2. transaction: This method give us the ability to control a transaction and performing either commit or rollback based on the situation.

3. executeSql: This method is used to execute actual SQL query.

Opening Database:
The openDatabase method takes care of opening a database if it already exists, this method will create it if it already does not exist.

To create and open a database, Following code:

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
```

Above method took following 5 paramters:
1. Database name
2. Version number
3. Text description
4. Size of databas
5. Creation callback

The last and 5th argument, creation callback will be called if the database is being created.

Executing queries:
To execute a query you use the database.transaction() function. This function needs a single argument, which is a function that takes care of actually executing the query as follows:

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
db.transaction(function (tx)
{
  tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
});
```

The above query will create a table called LOGS in 'mydb' database.

Note: You need to know that since November 18th, 2010, the W3C stopped supporting the Web SQL Database specification.

We have two types of databases supported by different browsers.
1 WebSql for Safari like browsers
2 IndexedDB for Mozilla, Chrome, Internet Explorer like browsers.

WebSql: This is a query based database like SQL server. You need to write queries to insert, update and delete the records like you did in SQL server. This database is used for the Safari browser.

IndexedDB : This database works on objects, we are not required to write queries for this database. It simply worked on to add, update and delete the objects. For this database, we will use a db.js wrapper file provided by aaron powell. We will use this database for all the browsers other than Safari.

HTML5 File / Hardware Access(Deeper integration with the Operating System)

FileSystem APIs:

Reading and manipulating files: File/Blob, FileList, FileReader
Creating and writing: Blob(), FileWriter
Directories and file system access: DirectoryReader, FileEntry/DirectoryEntry, LocalFileSystem

There are two versions of the FileSystem API. One for synchronous calls and another for asynchronous calls.
No longer do we need to download and install a given piece of software in order to use it. Simply a web browser and an internet connection gives us the ability to use any web application, anytime, anywhere, and on any platform.

The Filesystem API comes in two different versions. The asynchronous API, which is useful for normal applications, and the synchronous API, reserved for use with web workers.

Storage Types and Quotas
There are two types of storage
Temporary or persistent.

Temporary:
Data stored in a temporary filesystem may be deleted by the browser to free up space for another application. The advantage of using a temporary filesystem is that it is automatically granted by the browser; the user does not have to explicitly allow the browser to store data on their machine.

Persistent:
Data stored in a persistent filesystem is held in the browser until the user deletes it. You should use persistent storage if you plan to store data that is essential to your app. The slight downside of using a persistent filesystem is that the user has to grant you permission to store data

Handling Errors
Many of the methods provided by the FileSystem API support an error callback. Whilst this is technically optional, you will want to include an error handler so that you can debug your application if something goes wrong.

Sandboxing

Filesytems are sandboxed. This means that you can only access files that exist within your app's filesystem. The files in your app's filesystem are also protected from being accessed by any other applications. This is an important security feature.

How To Detect Browser Support

Example:

```
<body>
<script type='text/javascript'>
if (window.requestFileSystem)
{
alert("GreatFileSystemSupported");
}
else
{
alert("FileSystemNotSupported");
}
</script>
</body>
```

Step 1 - Getting Started

Your first step is to obtain access to the HTML5 Filesystem by requesting a LocalFile System object, using the window.requestFileSystem() global method.

```
window.requestFileSystem(type, size, successCallback, opt_error)
```

Step 2 - Working With Directories
Step 3 - Working With Files
Step 4 - Manipulating Files and Directories

HTML5 Accepting Speech Input in HTML5 Forms

The way that we interact with computers has changed dramatically over the past decade. Touch-screen devices and laptop trackpads have enabled a much more intuitive form of interaction than is achievable using a traditional mouse. These changes haven't been limited to just hardware.

Enabling Speech Input

Enabling support for speech input is as simple as adding an attribute to your <input> elements. The x-webkit-speech attribute will indicate to the browser that the user should be given the option to complete this form field using speech input.

```
<input type="text" x-webkit-speech>
```

Note: Google Chrome is the only browser that currently supports speech

When speech input is enabled the element will have a small microphone icon displayed on the right of the input. Clicking on this icon will launch a small tooltip to show that your voice is now being recorded. You can also start speech input by focussing the element and pressing Ctrl + Shift + . on Windows, or Command + Shift + . on Mac.

In JavaScript, you can test to see if an element has speech input enabled by examining it's webkitSpeech property. This is a boolean property and will therefore be set to true or false. You can override this property to enable or disable speech input on an element.

```
// Enable
element.webkitSpeech = true;

// Disable
element.webkitSpeech = false;
```

Detecting Browser Support
A simple way of checking if the user's browser supports speech input is to look for the webkitSpeech property on an <input> element. An example of how to do this is shown below.

Example:
```
<script type='text/javascript'>
if (document.createElement('input').webkitSpeech === undefined)
{
alert("Not supported");
}
else
{
alert("Supported!");
}
</script>
```

How Speech Recognition Works:
The browser relies on an external service to handle speech-to-text conversion. The recording of your voice is sent to this service which then analyses the audio and constructs a textual representation. The text is then sent back to the browser which populates the <input> element to complete the process.

HTML5 Device Orientation:

Using Device Orientation in HTML5
Device orientation is yet another cool feature of HTML5. Device orientation allows a device to detect its physical orientation with respect to gravity. If you've ever rotated a smart phone or tablet, and the screen has rotated in response. Orientation is measured using three angles – alpha, beta, and gamma.

Browser Compatibility
Before using the device orientation API, you need to make sure your browser supports it.

Example:
```
<script type='text/javascript'>
if (window.DeviceOrientationEvent)
{
alert("Browser supports DeviceOrientation");
}
else
{
alert("Sorry,doesn't support Device Orientation");
}
</script>
```

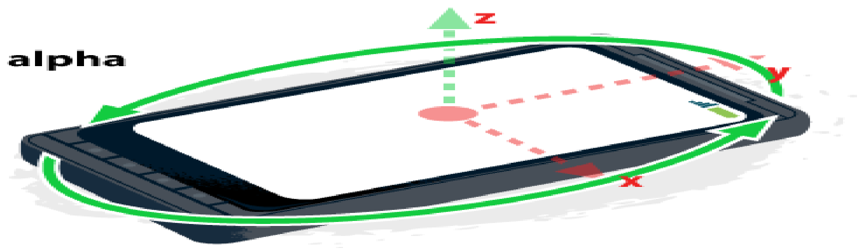
The deviceorientation Event
The deviceorientation event, which our code is listening for, is fired when the device orientation changes. When this event is fired, our event handler, deviceOrientationListener() is invoked. A DeviceOrientationEvent object is the only argument passed to our handler. The previously mentioned alpha, beta, and gamma angles are defined as properties of the DeviceOrientationEvent.

Example:
```
<!DOCTYPE html>
<html>
  <body>
    <canvas id="myCanvas" width="360" height="450" style="border:1px solid #d3d3d3;">
    </canvas>
    <script>
      if (window.DeviceOrientationEvent) {
        window.addEventListener("deviceorientation", deviceOrientationListener);
      } else {
        alert("Sorry, your browser doesn't support Device Orientation");
      }
    </script>
  </body>
</html>
```

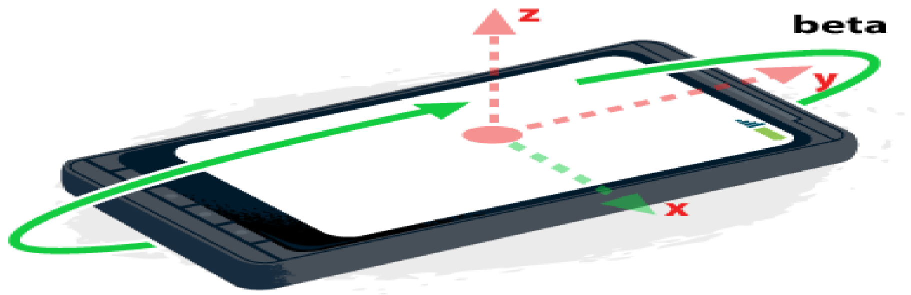
The Alpha, Beta, and Gamma Angles
Before explaining what each of the angles represents, we need to define the space in which they exist. The following image, courtesy of Mozilla, shows the 3D coordinate system used on mobile devices.

Alpha
The alpha angle represents rotation around the z-axis. Therefore, any rotation along the z-axis causes the alpha angle to change. The alpha angle can range between 0 and 360 degrees. Alpha is 0 when the top of the device is pointed directly to Earth's North Pole. The following image shows alpha rotation.

Alpha_Device_Orien

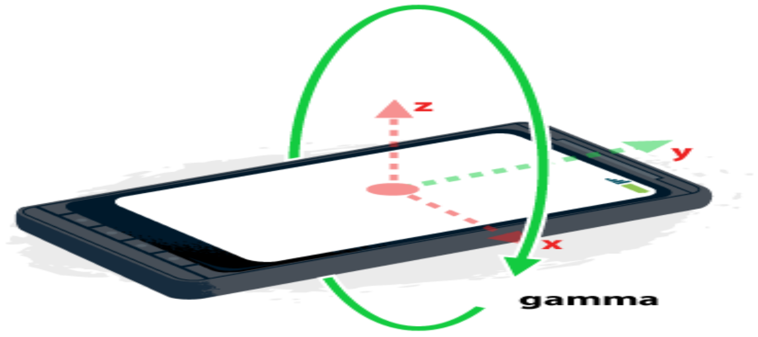


Beta
Rotation around the x-axis cause the beta angle to change. The range of beta is between -180 and 180 degrees. Beta is zero when the device is parallel to Earth's surface. An example of this would be lying on top of a table. An illustration of the beta angle is shown below.



Gamma

The gamma angle is associated with the y-axis. This angle ranges from -90 to 90 degrees, and is zero when the device is parallel to the Earth's surface. The gamma value changes when the device is rotated as shown in the following figure.



The deviceorientation Event Handler

The next step is to implement the handler for the deviceorientation event. This function is shown in the following code sample. This code begins by clearing the entire canvas. Next, a circle and rectangles are drawn according to the values of alpha, beta, and gamma.

Example:

```
<!DOCTYPE html>
<html>
 <body>
  <canvas id="myCanvas" width="360" height="450" style="border:1px solid #d3d3d3;">
  </canvas>
  <script>
   function deviceOrientationListener(event) {
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");

    ctx.clearRect(0, 0, c.width, c.height);
    ctx.fillStyle = "#FF7777";
    ctx.font = "14px Verdana";
    ctx.fillText("Alpha: " + Math.Round(event.alpha), 10, 20);
    ctx.beginPath();
    ctx.moveTo(180, 75);
    ctx.lineTo(210, 75);
    ctx.arc(180, 75, 60, 0, event.alpha * Math.PI / 180);
    ctx.fill();

    ctx.fillStyle = "#FF6600";
    ctx.fillText("Beta: " + Math.round(event.beta), 10, 140);
    ctx.beginPath();
```

```
      ctx.fillRect(180, 150, event.beta, 90);

      ctx.fillStyle = "#FF0000";
      ctx.fillText("Gamma: " + Math.round(event.gamma), 10, 270);
      ctx.beginPath();
      ctx.fillRect(90, 340, 180, event.gamma);
    }

    if (window.DeviceOrientationEvent) {
      window.addEventListener("deviceorientation", deviceOrientationListener);
    } else {
      alert("Sorry, your browser doesn't support Device Orientation");
    }
   </script>
  </body>
</html>
```

ADVANCED CASCADING STYLE SHEETS

CSS3&4

What is CSS?
Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.
OR
Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language.

Cascading:
Multiple styles can overlap in order to specify a range of style from a whole web site down to a unique element. Which style gets applied pertains to the rules of CSS cascading logic.

Style:
CSS deals specifically with the presentation domain of designing a web page (color, font, layout, etc).

Sheet:
Normally, CSS is a file separate from the HTML file –linked to the HTML file through its <head> (exceptions apply).

Features of CSS:
1. Flexibility

2. Codes Rendering
3. Accessibility
4. Easy Manage
5. Global Change
6. CSS Save a lot of time
7. Easy Maintenance
8. Inline Styles
9. Internal Style Sheets
10. External Style Sheets
11. Page Load Faster
12. Superior styles to HTML
13. Multiple Device Compatibility
14. Global web standards

CSS Versions:
Cascading Style Sheets, level 1 (CSS1) was came out of W3C as a recommendation in December 1996. This version describes the CSS language as well as a simple visual formatting model for all the HTML tags.

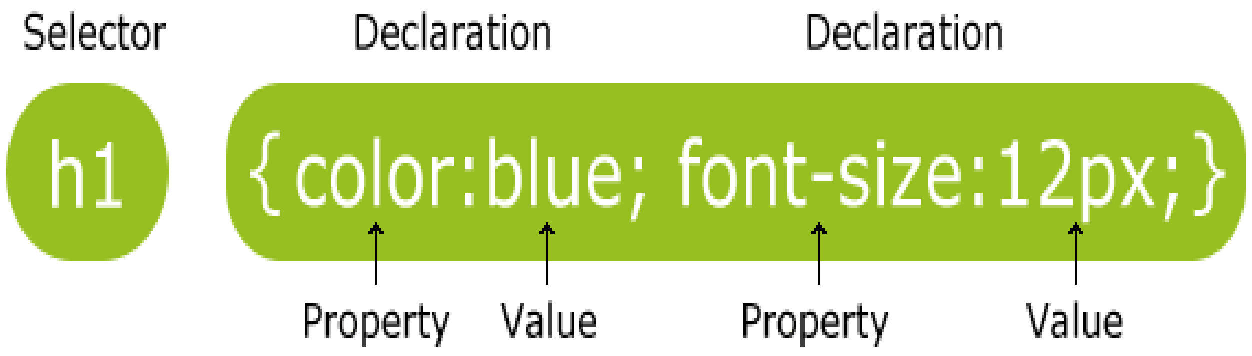
CSS2 was became a W3C recommendation in May 1998 and builds on CSS1. This version adds support for media-specific style sheets e.g. printers and aural devices, downloadable fonts, element positioning and tables.

CSS3 was became a W3C recommendation in May 2008 and builds on CSS2. This version adds support for responsive web designing and media queries.

CSS--Syntax
A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:
Selector:
A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
Property:
A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border, etc.
Value:
Values are assigned to properties. For example, color property can have the value either red or #F1F1F1 etc.

Syntax as follows:

HTML5&CSS3



CSS STRUCTURE

```
<html>
<head>
<style type="text/css">
{
---------------
---------------
}
</style>
</head>
<body>
---------------
---------------
</body>
</html>
```

CSS COMMENTS

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers.

A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
p
{
text-align:center;
/*This is another comment*/
color:#000000;
font-family:arial;
}
```

TYPES OF STYLESHEETS/Applying CSS

There are three ways to apply CSS to HTML.

- ✓ INLINE STYLES
- ✓ INTERNAL/Embedded STYLE SHEETS
- ✓ EXTERNAL STYLE SHEETS

Inline Styles:

We specify styles inside the tag in the body part. These styles will be applied only for that particular line. They look something like this:

Syntax:

<tag style=”property:value”>Some Text </tag>

Example:

```
<!doctype html>
<body>
<p style="color: red"> Naresh i Technologies</p>
<span style="color:blue"> This will make that text style in blue color..!!</span>
</body>
```

INTERNAL/Embedded STYLE SHEETS

If we specify the styles in our html file itself, then they are called as internal styles. These styles cannot be used in other files (i.e., if we want the same styles in other files, we should write them again in that other file) Embedded, or internal styles are used for the whole page. Inside the head tags, the style tags surround all of the styles for the page.

SYNTAX :

```
<!doctype html>
<html lang=”en-IN”>
<head>
<style type="text/css">
</style>
</head>
<body>
</body>
</html>
```

Example:

```
<!doctype html>
<html lang=”en-IN”>
<head>
<title>
CSS Internal Style Sheets
</title>
<style type=”text/csss”>
div
{
color:blue;font-size:30px;
}
</style>
</head>
<body>
<div>Welcome to CSS INTERNAL Style Sheets..</div>
<div>Welcome to CSS INTERNAL Style Sheets..</div>
</body>
```

HTML5&CSS3

EXTERNAL STYLE SHEETS
If we declare the styles outside our html file (as another file), then they are called External Styles. These styles can be reusable i.e., they can be used for more than one file. We save the external file consisting of styles with .css file extension. The changes made in external files will affect all the html files which are using those styles.

SYNTAX:

```
<head>
<link rel="stylesheet" href="Path of the resources"  type="text/css">
</head>
```

Example1:
Step 1
Prepare CSS file
Step 2
Prepare HTML file

Step 1

```
h1 {color: green}
h2 {color: #ff00ff}
```

Save with .css extension

Step 2

```
<head>
<link rel="stylesheet" href="one.css">
</head>
<body>
<h1>This is header 1</h1>
<h2>This is header 2</h2>
</body>
</html>
```

Save with .htm or .html Extension and run on any Major Web Browser..!!

CSS3 Features:
1. @font-face
2. Opacity
3. RGBA (red green blue alpha or Amber(LED driver for RGBA color mixing applications)
4. Border-radius
5. Box-shadow
6. Text-shadow
7. Gradient
8. Multiple background images
9. Transform
10. Transition

11. Multi-Column Layout
12. Styling Forms with Attribute Selectors
13. Wrapping Up
14. Box-sizing and Box-model
15. CSS3 Selectors
16. Tooltip with CSS3
17. Simple CSS3 rounded Corners with Support for IE

CSS3-MODULES

Module Focus	Description
2D Transforms	Provides for manipulation of content in two dimensions, such as rotating, scaling, and skewing objects.
3D Transforms	Extends 2D Transforms to manipulate elements in a three-dimensional space.
Animations	Introduces the ability to modify CSS property values over time, such as position or color, to create animated layouts.
Backgrounds and Borders	Introduces multiple backgrounds and a variety of background properties for positioning and sizing. Some interesting new border properties allow for styling borders with images, shadows, and more.
Behavioral Extensions	Defines components that can be attached to elements on a page to enhance their functionality.
Box Model	Defines standard boxes, including float, margins, over flow, and padding.
Color	Defines the color units supported in CSS as

	well as a few color properties like color and opacity. It mostly documents CSS2 but includes some new ideas like the currentColor keyword.
Fonts	Defines the standard font properties but introduces new font decoration features like font-effect, font-smooth, and fontemphasize, which are not supported by any browsers as of yet.
Generated Content for Paged Media	Defines the management of generated content for print output, including crop mark indication, header/footer handling, and much more.
Generated and Replaced Content	Defines the management of generated content, including inserted content, counters, footnotes, and so on.
Grid Positioning	Defines the use of grid-based layouts with standard CSS sizing and positioning properties.
Hyperlink Presentation	Defines the presentation and effects for hyperlinks.
Line Layout	Defines line-formatting properties such as vertical line alignment, line height, and first line and first letter visual effects.
Lists	Defines the handling of lists, including marker styles and some aspects of counters.
Marquee	Defines properties to create animated content, employing a “marquee” effect similar to the nonstandard HTML tag of the same name

	(<marquee>). See the entries for marquee-direction, marqueeplay-count, marquee-speed, and marquee-style later in the chapter.
Media Queries	Defines CSS syntax for applying different style rules based upon media or device characteristics, such as width or color support, avoiding the use of JavaScript to reapply style. See the section “Media Queries” later in the chapter for syntax and examples.
Multi-column Layout	Defines how to flow text into many columns.
Namespaces	Defines syntax to allow the disambiguation of elements from different markup languages found within the same document for styling purposes.
Paged Media	Defines how pagination is per formed, particularly with print output.
Presentation Levels	Defines the concept of applying presentation levels to style elements in different manners depending upon the situation.
Ruby	Defines the CSS-handling aspects of Ruby texts, which are used to provide pronunciation or alternate readings in East Asian languages.
Selectors	Defines the various selectors for standard CSS1 and CSS2 and introduces numerous complex tree- and attribute-specific syntax.
Speech	Continues prior support of aural style sheets and introduces new values to improve pronunciation like phonemes, but also seems

	to simply rename features. For example, stress becomes voicestress, pitch becomes voice-pitch, and volume becomes voice-volume.
Template Layout	Defines a layout grid for positioning and alignment of Web applications or documents. Provides for a template-like system that has some characteristics similar to classic markup tables.
Text	Defines text manipulation, including alignment, line breaking, justification, text decoration, text transformation, and whitespace handling.
Transitions	Defines how property changes can be applied to CSS rules over a specified duration of time. Useful for animating simple visual changes.
User Interface	Defines properties and selectors useful for styling user inter faces, such as cursor and navigation handling, as well as the current state of elements, such as valid versus invalid, active versus disabled, and so on.
Web Fonts	Codifies and improves upon downloadable fonts, which have long been supported in Internet Explorer. See the section “Web Fonts” and Appendix B for more information.
Values and Units	Expands the absolute and relative units of measure, including significant changes to support animation and aural changes with time (s and ms) and angle (deg and rad) values.

CSS3 Browser Support:
1. Internet Explorer 9 requires the prefix -ms-. (Microsoft seem)
2. Firefox requires the prefix -moz-.
3. Chrome and Safari requires the prefix -webkit-.
4. Opera requires the prefix -o-.

CSS3 Backgrounds
CSS3 contains several new background properties, which allow greater control of the background element.

Property	**Description**
background-size	Specifies the size of the background images
background-origin	Specifies the positioning area of the background images
background-clip	Specifies the painting area of the background images

background-size:
The background-size property specifies the size of the background image. Before CSS3, the background image size was determined by the actual size of the image. In CSS3 it is possible to specify the size of the background image, which allows us to re-use background images in different contexts.

Syntax
background-size: length|percentage|cover|contain;

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
body
{
background:url(img_flwr.gif);
background-size:80px 60px;
-moz-background-size:80px 60px;
background-repeat:no-repeat;
padding-top:40px;
}
</style>
</head>
<body>
```

```
<p>
CSS-Image
</p>
<p>Original image: <img src="img_flwr.gif" alt="Flowers" width="250" height="200" /></p>
</body>
</html>
```

Example:(Stretch the background-image)

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
background:url(img_flwr.gif);
background-size:100% 100%;
-moz-background-size:100% 100%;
background-repeat:no-repeat;
}
</style>
</head>
<body>
<div>

With CSS you define the colors and sizes in "styles". Then as you write your documents you refer to the styles. Therefore: if you change a certain style it will change the look of your entire site.

Another big advantage is that CSS offers much more detailed attributes than plain HTML for defining the look and feel of your site.

   CSS stands for Cascading Style Sheets
   Styles define how to display HTML elements
   Styles were added to HTML 4.0 to solve a problem
   External Style Sheets can save a lot of work
   External Style Sheets are stored in CSS files

</div>
<p></p>
<div>
```

With CSS you define the colors and sizes in "styles". Then as you write your documents you refer to the styles. Therefore: if you change a certain style it will change the look of your entire site.

Another big advantage is that CSS offers much more detailed attributes than plain HTML for defining the look and feel of your site.

```
   CSS stands for Cascading Style Sheets
   Styles define how to display HTML elements
   Styles were added to HTML 4.0 to solve a problem
   External Style Sheets can save a lot of work
   External Style Sheets are stored in CSS files
</div>
</body>
</html>
```

CSS3 Multiple Background Images
CSS3 allows you to use several background images for an element.

Syntax:
```
background-image:url(img1),url(img2);
```

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
body
{
background-image:url(img_flwr.gif),url(xmas-tree.gif);
}
</style>
</head>
<body>
</body>
</html>
```

background-clip: This property specifies the painting area of the background.

Syntax:

background-clip: border-box|padding-box|content-box;

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style>
div
{
width:300px;
height:300px;
padding:50px;
background-color:lightblue;
background-clip:content-box;
border:4px solid #FF0000;
text-align:justify;
}
</style>
</head>
<body>
<div>
Cascading Style Sheets were developed as a means for creating a consistent approach to providing style information for web documents. CSS has various levels and profiles. Each level of CSS builds upon the last. CSS3 has all advance features.
<p> Cascading Style Sheets were developed as a means for creating a consistent approach to providing style information for web documents. typically adding new features and typically denoted as CSS 1, CSS 2, CSS 3, and CSS 4.  </p>
</div>
</body>
</html>
```

Note: border porperty change to dotted and observe output.

background-origin Property:
The background-origin property specifies, what the background-position property should be relative to.

Syntax
background-origin: padding-box|border-box|content-box;

Example:

HTML5&CSS3

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style>
div
{
border:2px solid red;padding:30px;
background-image:url('chrome.png');
background-repeat:no-repeat;
background-position:left;
background-origin:border-box;
}
</style>
</head>
<body>
<div>
Google Chrome is a browser that combines a minimal design with sophisticated technology
to make the Web faster, safer, and easier. Use one box for everything type in the address
bar and get suggestions for both search and Web pages.
</div>
</body>
</html>
```

CSS3 Borders

With CSS3, you can create rounded borders, add shadow to boxes, and use an image as a border - without using a design program, like Photoshop. following border properties:

1. border-radius
2. box-shadow
3. border-image

CSS3 Rounded Corners

Adding rounded corners in CSS2 was tricky. We had to use different images for each corner.
In CSS3, creating rounded corners is easy.
In CSS3, the border-radius property is used to create rounded corners:

New Border Properties

Property	Description
border-image	A shorthand property for setting all the border-image-*properties

border-radius A shorthand property for setting all the four border-*-radius properties
box-shadow   Attaches one or more drop-shadows to the box

border-radius: It is a shorthand property for setting the four border-*-radius properties

Syntax
border-radius: 1-4 length|% /px 1-4 length|%;

JavaScript syntax:
object.style.borderRadius="5px"

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
border:2px solid #b2b2b2;
padding:10px 40px;
background:#ccddcc;
width:300px;
border-radius:25px;
</style>
</head>
<body>
<div>The border-radius property allows you to add rounded corners to elements.</div>
</body>
</html>
```

CSS3 Box Shadow
The box-shadow property attaches one or more drop-shadows to the box.

Syntax
box-shadow: h-shadow v-shadow blur spread-color inset;

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
```

```
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:300px;height:100px;
background-color:red;
box-shadow: 10px 10px 5px #990099;
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

Example

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:300px;height:100px;
background-color:yellow;
box-shadow: 10px 10px 50px 20px pink inset
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

CSS3 Border Image

The border-image property is a shorthand property for setting the border-image-source, border-image-slice, border-image-width, border-image-outset and border-image-repeat properties.

Syntax

```
border-image: source slice width outset repeat;
```

Note: The border-image property is supported in Firefox and Chrome.
Note: Opera supports an alternative, the -o-border-image property.
Note: Safari supports an alternative, the -webkit-border-image property.

Value	**Description**
border-image-source	The path to the image to be used as a border
border-image-slice	The inward offsets of the image-border
border-image-width	The widths of the image-border
border-image-outset	The amount by which the border image area extends beyond the border box
border-image-repeat	Whether the image-border should be repeated,rounded or stretched

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
border:15px solid transparent;
width:250px;
padding:10px 20px;
}
#round
{
border-image:url(border.png) 30 30 round;
}
#stretch
{
border-image:url(border.png) 30 30 stretch;
}
</style>
</head>
<body>
<div id='round'>Here, the image is tiled (repeated) to fill the area.</div>
<br>
<div id='stretch'>Here, the image is stretched to fill the area.</div>
</body>
</html>
```

CSS3 Text Effects

New Text Properties

Property	Description
hanging-punctuation	Specifies whether a punctuation character may beplacedoutside the line box
punctuation-trim	Specifies whether a punctuation character should be trimmed
text-emphasis	Applies emphasis marks, and the foreground color of the emphasis marks
text-justify	Specifies the justification method used when text-align is "justify"
text-outline	Specifies a text outline
text-overflow	Specifies what should happen when text overflows the containing element
text-shadow	Adds shadow to text
text-wrap	Specifies line breaking rules for text
word-break	Specifies line breaking rules for non-CJK scripts
word-wrap	Allows long, unbreakable words to be broken and wrap to the next line

CSS3 hanging-punctuation Property:

The hanging-punctuation property specifies whether a punctuation mark may be placed outside the line box at the start or at the end of a full line of text.

Note: The hanging-punctuation property is not supported in any of the major browsers.

Syntax

hanging-punctuation: none|first|last|allow-end|force-end;

Value	**Description**
first	Punctuation may hang outside the start edge of the first line
last	Punctuation may hang outside the end edge of the last line

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
p
```

```
{
hanging-punctuation:first;
}
</style>
<body>
<p> NoBrowserSupport</p>
</body>
</html>
```

CSS3 punctuation-trim Property

The punctuation-trim property specifies whether a punctuation character should be trimmed if it appears at the start or end of a line, or adjacent to another fullwidth punctuation character.

Syntax

punctuation-trim: none|start|end|allow-end|adjacent;

JavaScript syntax:

object.style.punctuationTrim="start";

Note: The punctuation-trim property is not supported in any of the major browsers.

Value	Description
none	Do not trim opening or closing punctuation marks
start	Trim opening punctuation at the beginning of each line
end	Trim closing punctuation at the end of each line

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
p
{
punctuation-trim:start;
}
</style>
<body>
<p> NoBrowserSupport</p>
</body>
</html>
```

CSS3 text-emphasis Property
The text-emphasis property is a shorthand for setting text-emphasis-style and text-emphasis-color in one declaration.

Note: The text-emphasis property is not supported in any of the major browsers.
Syntax
text-emphasis: text-emphasis-style text-emphasis-color;

JavaScript syntax:
object.style.textEmphasis="filled blue"

Value	**Description**
text-emphasis-style	Applies emphasis marks to the element's text
text-emphasis-color	Defines the foreground color of the emphasis marks

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
p
{
text-align:justify;
text-justify:inter-word;
}
</style>
<body>
<p> NoBrowserSupport</p>
</body>
</html>
```

CSS3 text-justify Property
This property specifies how justified text should be aligned and spaced.

Syntax
text-justify: auto|inter-word|inter-ideograph|inter-cluster|distribute|kashida|trim;

JavaScript syntax:
object.style.textJustify="inter-word"

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
div
{
text-align:justify;
text-justify:inter-word;
}
</style>
</head>
<body>
<h1>CSS3 text-justify Example-Gmail</h1>
<div> Custom themes, The number of available themes has increased from 35 to...infinity. Select your own image to use as a custom theme, or choose from a selection of featured photos.  The number of available themes has increased from 35 to...infinity. Select your own image to use as a custom theme, or choose from a selection of featured photos.</div>
</body>
</html>
```

CSS3 text-outline Property:
The text-outline property specifies a text outline.

Note: The text-outline property is not supported in any of the major browsers.

Syntax

```
text-outline: thickness blur color;
```

JavaScript syntax:

```
object.style.textOutline="2px 2px #ff0000"
```

Value	**Description**
thickness	Required. The thickness of the outline
blur	Optional. The blur radius of the outline
color	Required. The color of the outline.

Example

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
```

```
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p
{
text-outline: 2px 2px #ff0000;
}
</style>
</head>
<body>
<h1>CSS3 text-outline Example-Gmail</h1>
<p> Custom themes, The number of available themes has increased from 35 to...infinity. Select your own image to use as a custom theme, or choose from a selection of featured photos. </p>
</body>
</html>
```

CSS3 text-overflow Property:
The text-overflow property specifies what should happen when text overflows the containing element.

Note: The text-overflow property is supported in all major browsers.

Syntax
text-overflow: clip|ellipsis|string;

JavaScript syntax:
object.style.textOverflow="ellipsis"

Value	**Description**
clip	Clips the text
ellipsis	Render an ellipsis ("...") to represent clipped text
string	Render the given string to represent clipped text

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style>
div.test
{
```

```
white-space:nowrap;
width:12em;
overflow:hidden;
border:1px solid #ff0000;
}
</style>
</head>
<body>
<p>This div uses "text-overflow:ellipsis":</p>
<div class="test" style="text-overflow:ellipsis;">This is some long text that will not fit in the box</div>
<p>This div uses "text-overflow:clip":</p>
<div class="test" style="text-overflow:clip;">This is some long text that will not fit in the box</div>
</body>
</html>
```

Text-Shadow:
The text-shadow property applies shadow to text. You specify the horizontal shadow, the vertical shadow, the blur distance, and the color of the shadow:

Note: The text-shadow property is supported in all major browsers, except Internet Explorer.

Syntax
text-shadow: h-shadow v-shadow blur color;

JavaScript syntax:
object.style.textShadow="2px 2px #ff0000"

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
h1
{
text-shadow: 5px 5px 5px #FF0011;
}
</style>
</head>
```

```
<body>
<h1>NareshTechnologies!</h1>
</body>
</html>
```

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
h1
{
text-shadow:-10px -10px red
}
</style>
</head>
<body>
<h1>NareshTechnologies!</h1>
</body>
</html>
```

CSS3 text-wrap Property
The text-wrap property specifies line breaking rules for text.

Note:The text-wrap property is not supported in any of the major browsers.

Syntax
text-wrap: normal|none|unrestricted|suppress;

JavaScript syntax:
object.style.textWrap="none"

Value	Description
normal	Lines may break only at allowed break points
none	Lines may not break. Text that does not fit in the element, overflows it
unrestricted	Lines may break between any two characters
suppress	Line breaking is suppressed within the element

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
```

```
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style>
p.test1
{
width:11em;
border:1px solid #ff0000;
text-wrap:none;
}
</style>
</head>
<body>
<p class="test1"> This paragraph contains some text. This line should not breake or wrap to the next line.</p>
</body>
</html>
```

CSS3 word-break Property
The word-break property specifies line breaking rules for non-CJK scripts.
Note: Tip: CJK scripts are Chinese, Japanese and Korean ("CJK") scripts.
Note: The word-break property is supported in all major browsers, except Opera.

Syntax
word-break: normal|break-all|hyphenate;

JavaScript syntax:
object.style.wordBreak="hyphenate"

Value	**Description**
normal	Breaks non-CJK scripts according to their own rules
break-all	Lines may break between any two characters for non-CJK scripts
hyphenate	Words may be broken at an appropriate hyphenation point

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p.test1
{
```

```
width:11em;
border:1px solid #ff0000;
word-break:hyphenate;
}
p.test2
{
width:11em;
border:1px solid #0000ff;
word-break:break-all;
}
</style>
</head>
<body>
<p class="test1"> This paragraph contains some text. This line will-break-at-hyphenates.</p>
<p class="test2"> This paragraph contains some text: The lines will break at any character.</p>
</body>
</html>
```

CSS3 word-wrap Property
The word-wrap property allows long words to be able to be broken and wrap onto the next line

Note: The word-wrap property is supported in all major browsers.

JavaScript syntax:
object.style.wordWrap="break-word"

Syntax
word-wrap: normal|break-word;

Value	**Description**
normal	Break words only at allowed break points
break-word	Allows unbreakable words to be broken

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
p.test
```

```
{
width:11em;
border:1px solid #00dd00;
word-wrap:break-word;
}
</style>
</head>
<body>
<p class="test"> This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.</p>
</body>
</html>
```

CSS3 Fonts:

In CSS3 the font properties are inserted:
1. @font-face
2. font-size-adjust
3. font-stretch

@font-face (CSS3 Web Fonts)
The CSS3 @font-face Rule:Your "own" fonts are defined in the CSS3 @font-face rule. Before CSS3, web designers had to use fonts that were already installed on the user's computer. With CSS3, web designers can use whatever font they likes.

Note:
Firefox, Chrome, Safari, and Opera support fonts of type .ttf (True Type Fonts) and .otf (OpenType Fonts).

Note:
Internet Explorer 9+ supports the new @font-face rule, but it only supports fonts of type .eot (Embedded OpenType).

Syntax:

```
@font-face
{
font-properties
}
```

Different Font Formats
TrueType Fonts (TTF)
TrueType is a font standard developed in the late 1980s, by Apple and Microsoft.

OpenType Fonts (OTF)
OpenType is a format for scalable computer fonts. It was built on TrueType, and is a registered trademark of Microsoft.

The Web Open Font Format (WOFF)
WOFF is a font format for use in web pages. It was developed in 2009, and is now a W3C Recommendation. All Browsers Compatable

SVG Fonts/Shapes
SVG fonts allow SVG to be used as glyphs when displaying text.

Embedded OpenType Fonts (EOT)
EOT fonts are a compact form of OpenType fonts designed by Microsoft for use as embedded fonts on web pages.

CSS3 Font Descriptors
The following table lists all the font descriptors that can be defined inside the @font-face rule:

Name	**Values**	**Description**
font-family	name	Defines a name for the font
src	URL	Defines the URL of the font file
font-stretch	normal	Defines how the font should be stretched
font-style	normal	
	italic	
	oblique	Defines how the font should be styled.
font-weight	normal	Defines the boldness of the font.
	bold	

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
@font-face
{
font-family: myFirstFont;
src: url('Sansation_Light.ttf')
}
div
```

```
{
font-family:myFirstFont;
}
</style>
</head>
<body>
<div>
With CSS3, We can declare user defined fonts as per the requirements.
</div>
</body>
</html>
```

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
@font-face
{
font-family: myFirstFont;
src: url('Sansation_Light.ttf')
}
@font-face
{
font-family: myFirstFont;
src: url('Sansation_Bold.ttf')
font-weight:bold;
}
div
{
font-family:myFirstFont;
}
</style>
</head>
<body>
<div>
With CSS3, websites can <b>finally</b> use fonts other than the pre-selected "web-safe"
fonts.
</div>
</body>
</html>
```

CSS3 font-size-adjust Property:
The font-size-adjust property gives you better control of the font size when the first selected font is not available. When a font is not available, the browser uses the second specified font. This could result in a big change for the font size. To prevent this, use the font-size-adjust property.

Syntax
font-size-adjust: number|none|inherit;

Value	**Description**
number	Defines the aspect value to use
none	Default value. No font size adjustment
inherit	nherits the font size adjustment from parent elements

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
.divVerdana {font-family: verdana;}
.divTimes {font-family: 'times new roman';}
#div1 {font-size-adjust:0.58;}
#div2 {font-size-adjust:0.58;}
</style>
</head>
<body>
<p>Two divs using the same font-size-adjust property:</p>
<div id="div1" class="divVerdana">
You control the font size better with the font-size-adjust property.</div>

<div id="div2" class="divTimes">
You control the font size better with the font-size-adjust property</div>

<p>The same two divs without the font-size-adjust property:</p>
<div class="divVerdana">
You control the font size better with the font-size-adjust property.</div>

<div class="divTimes">
You control the font size better with the font-size-adjust property.</div>
</body>
```

```
</html>
```

CSS3 font-stretch Property
The font-stretch property allows you to make text wider or narrower.

Note: None of the major browsers support the font-stretch property.

Syntax
font-stretch:wider|narrower|ultra-condensed|extra-condensed|condensed|semi-condensed|normal|semi-expanded|expanded|extra-expanded|ultra-expanded|inherit;

Value	**Description**
wider	Makes the text wider
narrower	Makes the text narrower
ultra-condensed	Makes the text as narrow as it gets
extra-condensed	Makes the text narrower than condensed, but not as narrow as ultra-condensed
condensed	Makes the text narrower than semi-condensed, but not as narrow as extra-condensed
semi-condensed	Makes the text narrower than normal, but not as narrow as condensed normal Default value. No font stretching
semi-expanded	Makes the text wider than normal, but not as wide as expanded
expanded	Makes the text wider than semi-expanded, but not as wide as extra-expanded
extra-expanded	Makes the text wider than expanded, but not as wide as ultra-expanded
ultra-expanded	Makes the text as wide as it gets
inherit	Inherits the font stretching from parent elements

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
#div1 {font-stretch:condensed;}
#div2 {font-stretch:expanded;}
</style>
</head>
<body>
<div>
```

```
Some normal text in a div element.
</div>
<div id="div1">
Some condensed text in a div element.
</div>
<div id="div2">
Some expanded text in a div element.
</div>
</body>
</html>
```

CSS3 Transforms:

It has the following three properties:
1. transform
2. transform-origin
3. transform-style

Transform: With CSS3 transform, we can move, scale, turn, spin, and stretch elements.

How Does it Work?
A transform is such a property of CSS3, which is used for changing the actual form of the element. With this feature of CSS3 You can change the shape, size and position of an element.

2D Transform Methods

Function	**Description**
matrix(n,n,n,n,n,n)	Defines a 2D transformation, using a matrix of six values
translate(x,y)	Defines a 2D translation, moving the element along the X- and the Y-axis
translateX(n)	Defines a 2D translation, moving the element along the X-axis
translateY(n)	Defines a 2D translation, moving the element along the Y-axis
scale(x,y)	Defines a 2D scale transformation, changing the elements width and height
scaleX(n)	Defines a 2D scale transformation, changing the element's width
scaleY(n)	Defines a 2D scale transformation, changing the element's height
rotate(angle)	Defines a 2D rotation, the angle is specified in the parameter
skew	Defines a 2D skew transformation along the X- and the Y- axis x-angle,y-angle)
skewX(angle)	Defines a 2D skew transformation along the X-axis
skewY(angle)	Defines a 2D skew transformation along the Y-axis

A 2D Transform is such an amazing feature of CSS3 Transform, which is used for the following methods.
1. translate()

2. rotate()
3. scale()
4. skew()
5. matrix()

translate() Method:
With the help of this method you can move your object depending on its parameter. Two type of parameter you can pass in this method one is from left (x-axis) and the another is from top (y-axis).

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:80px;
height:70px;
background-color:blue;
border:2px solid #ff0000;
color:red;
}
div#div1
{
transform:translate(40px,80px);
}
</style>
</head>
<body>
<div>KSNareshITu</div>
<div id='div1'>KSNareshITu</div>
</body>
</html>
```

rotate() Method:
With the help of this method you can rotate your object depending on its value. Two type of value you can pass in this method one is positive (for clockwise rotation) and the another one is negative (for counter-clockwise rotation).

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:75px;
background-color:red;
border:1px solid black;

-ms-transform:rotate(30deg); /* IE 9 */
-moz-transform:rotate(30deg); /* Firefox */
-webkit-transform:rotate(30deg); /* Safari and Chrome */
-o-transform:rotate(30deg); /* Opera */
}
div#div2
{
transform:rotate(30deg);
}
</style>
</head>
<body>
<div>Hello. This is a DIV element.</div>
<div id="div2">Hello. This is a DIV element.</div>
</body>
</html>
```

scale() Method:
With the help of this method you can increase or decrease your object size depending on its value passed in the parameter. Two types of value you can pass in the parameter, one is for width (x-axis) and the another one for height (y-axis).

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
```

```
{
width:100px;
height:75px;
background-color:red;
border:1px solid black;
}
div#div2
{
margin:100px;
transform:scale(2,4);

-ms-transform:scale(2,4); /* IE 9 */
-moz-transform:scale(2,4); /* Firefox */
-webkit-transform:scale(2,4); /* Safari and Chrome */
-o-transform:scale(2,4); /* Opera */
}
</style>
</head>
<body>
<div>Hello. This is a DIV element.</div>
<div id="div2">Hello. This is a DIV element.</div>
</body>
</html>
```

skew() Method:
With the help of this method you can change the angle of your object depending on its value passed in the parameter. Two types of value you can pass in the parameter, one is for horizontal (x-axis) and the another one for vertical (y-axis).

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:75px;
background-color:red;
border:1px solid black;
}
```

```
div#div2
{
transform:skew(30deg,20deg);
-ms-transform:skew(30deg,20deg); /* IE 9 */
-moz-transform:skew(30deg,20deg); /* Firefox */
-webkit-transform:skew(30deg,20deg); /* Safari and Chrome */
-o-transform:skew(30deg,20deg); /* Opera */

}
</style>
</head>
<body>
<div>Hello. This is a DIV element.</div>
<div id="div2">Hello. This is a DIV element.</div>
</body>
</html>
```

matrix() Method:
With the help of this method you can change transform at one time, has been defined above. Six types of value you can pass in the parameter.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:75px;
background-color:red;
border:1px solid black;
}
div#div2
{
transform:matrix(0.866,0.5,-0.5,0.866,0,0);
-ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* IE 9 */
-moz-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Firefox */
-webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Safari and Chrome */
-o-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Opera */
}
```

```
</style>
</head>
<body>
<div>Hello. This is a DIV element.</div>
<div id="div2">Hello. This is a DIV element.</div>
</body>
</html>
```

CSS3 transform Property:
The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.

Syntax
transform: none|transform-functions;

Note:
1. The transform property is not supported in any browser.
2. Internet Explorer supports an alternative, the -ms-transform property (2D transforms only).
3. Firefox supports an alternative, the -moz-transform property (2D transforms only).
4. Opera supports an alternative, the -o-transform property (2D transforms only).
5. Safari and Chrome support an alternative, the -webkit-transform property (3D and 2D transforms).

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:200px;
height:100px;
background-color:lightblue;
transform:rotate(10deg);
color:red;
-ms-transform:rotate(10deg); /* IE 9 */
-moz-transform:rotate(10deg); /* Firefox */
-webkit-transform:rotate(10deg); /* Safari and Chrome */
-o-transform:rotate(10deg); /* Opera */
}
```

```
</style>
</head>
<body>
<div>KSNareshITu</div>
</body>
</html>
```

CSS3 transform-origin Property

The transform-origin property allows you to change the position on transformed elements. 2D transformed element can change the x- and y-axis of the element. 3D transformed element can also change the z-axis of the element.

Syntax

transform-origin: x-axis y-axis z-axis;

Property Value	**Description**
x-axis	Defining where the view is placed at the x-axis. Possible values: left center right length %
y-axis	Defining where the view is placed at the y-axis. Possible values: top center bottom length %
z-axis	Defining where the view is placed at the z-axis. Possible values: length

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
#div1
{
position: relative;
height: 200px;
width: 200px;
margin: 100px;
padding:10px;
border: 1px solid black;
}
#div2
{
padding:50px;
position: absolute;
border: 1px solid black;
background-color: red;
transform: rotate(45deg);
transform-origin:20% 40%;
-ms-transform: rotate(45deg); /* IE 9 */
-ms-transform-origin:20% 40%; /* IE 9 */
-webkit-transform: rotate(45deg); /* Safari and Chrome */
-webkit-transform-origin:20% 40%; /* Safari and Chrome */
-moz-transform: rotate(45deg); /* Firefox */
-moz-transform-origin:20% 40%; /* Firefox */
-o-transform: rotate(45deg); /* Opera */
-o-transform-origin:20% 40%; /* Opera */
}
</style>
</head>
<body>
<div id="div1">
  <div id="div2">HELLO</div>
</div>
</body>
</html>
```

CSS3 transform-style Property
The transform-style property specifies how nested elements are rendered in 3D space.

Note:
1. This property must be used together with the transform property.
2. The transform-style property is not supported in any browser.
3. Chrome and Safari support an alternative, the -webkit-transform-style property.

Syntax
transform-style: flat|preserve-3d;

Property Value	**Description**
flat	The child elements will NOT preserve its 3D position

preserve-3d The child elements will preserve its 3D position

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
#div1
{
position: relative;
height: 200px;
width: 200px;
margin: 100px;
padding:10px;
border: 1px solid black;
}

#div2
{
padding:50px;
position: absolute;
border: 1px solid black;
background-color: red;
transform: rotateY(60deg);
transform-style: preserve-3d;
-webkit-transform: rotateY(60deg); /* Safari and Chrome */
-webkit-transform-style: preserve-3d; /* Safari and Chrome */
}
#div3
{
padding:40px;
position: absolute;
border: 1px solid black;
background-color: yellow;
transform: rotateY(80deg);
-webkit-transform: rotateY(-60deg); /* Safari and Chrome */
}
</style>
</head>
<body>
<div id="div1">
```

```
  <div id="div2">HELLO
          <div id="div3">YELLOW</div>
  </div>
</div>
</body>
</html>
```

CSS3 3D Transforms

3D Transforms

CSS3 allows you to format your elements using 3D transforms. A transform is such a property of CSS3, which is used for changing the actual form of the element. With this feature of CSS3 You can change the shape, size and position of an element. A 3D Transform is such an amazing feature of CSS3 Transform, which is used for the following methods.
rotateX()
rotateY()

Note:
1. Internet Explorer and Opera does not yet support 3D transforms (They support only 2D transforms).
2. Firefox requires the prefix -moz-.
3. Chrome and Safari requires the prefix -webkit-.

Transform Properties

Property	**Description**
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
backface-visibility	Defines whether or not an element should be visible when not facing the screen

3D Transform Methods

Function	Description
matrix3d	Defines a 3D transformation, using a 4x4 matrix of 16 values (n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)
translate3d(x,y,z)	Defines a 3D translation
translateX(x)	Defines a 3D translation, using only the value for the X-axis
translateY(y)	Defines a 3D translation, using only the value for the Y-axis
translateZ(z)	Defines a 3D translation, using only the value for the Z-axis
scale3d(x,y,z)	Defines a 3D scale transformation
scaleX(x)	Defines a 3D scale transformation by giving a value for the X-axis

scaleY(y)	Defines a 3D scale transformation by giving a value for the Y-axis
scaleZ(z)	Defines a 3D scale transformation by giving a value for the Z-axis
rotate3d	Defines a 3D rotation (x,y,z,angle)
rotateX(angle)	Defines a 3D rotation along the X-axis
rotateY(angle)	Defines a 3D rotation along the Y-axis
rotateZ(angle)	Defines a 3D rotation along the Z-axis
perspective(n)	Defines a perspective view for a 3D transformed element

rotateX() Method:
With the help of this method you can rotate your object towards X-axis at given degree.

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:75px;
background-color:blue;
border:1px solid black;
color:red;
}
div#div2
{
transform:rotateX(120deg);
-webkit-transform:rotateX(120deg); /* Safari and Chrome */
-moz-transform:rotateX(120deg); /* Firefox */
}
</style>
</head>
<body>
<div>Hello. This is a DIV element.</div>
<div id="div2">Hello. This is a DIV element.</div>
</body>
</html>
```

rotateY() Method:
With the help of this method you can rotate your object towards Y-axis at given degree.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:75px;
background-color:blue;
border:1px solid black;
color:red;
}
div#div2
{
transform:rotateY(130deg);
-webkit-transform:rotateY(130deg); /* Safari and Chrome */
-moz-transform:rotateY(130deg); /* Firefox */
}
</style>
</head>
<body>
<div>Hello. This is a DIV element.</div>
<div id="div2">Hello. This is a DIV element.</div>
</body>
</html>
```

Transform Properties

CSS3 perspective Property

The perspective property defines how many pixels a 3D element is placed from the view. This property allows you to change the perspective on how 3D elements are viewed. When defining the perspective property for an element, it is the CHILD elements that get the perspective view, NOT the element itself.

Note:

1. The perspective property only affects 3D transformed elements!
2. Use this property together with the perspective-origin property, which allows you to change the bottom position of 3D elements.
3. The perspective property is not supported in any browser.

4. Chrome and Safari support an alternative, the -webkit-perspective property.

Syntax
perspective: number|none;

Property Value	**Description**
number	How many pixels the element is placed from the view
none	Default value. Same as 0. The perspective is not set

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
#div1
{
position: relative;
height: 150px;
width: 150px;
margin: 50px;
padding:10px;
border: 1px solid black;
perspective:150;
-webkit-perspective:150; /* Safari and Chrome */
}
#div2
{
padding:50px;
position: absolute;
border: 1px solid black;
background-color: red;
transform: rotateX(45deg);
-webkit-transform: rotateX(45deg); /* Safari and Chrome */
}
</style>
</head>
<body>
<div id="div1">
  <div id="div2">KSNareshITu</div>
</div>
 </body>
```

</html>

CSS3 perspective-origin Property:
The perspective-origin property defines where a 3D element is based in the x- and the y-axis. This property allows you to change the bottom position of 3D elements. When defining the perspective-origin property for an element, it is the CHILD elements that are positioned, NOT the element itself.

Note:
This property must be used together with the perspective property, and only affects 3D transformed elements!

Note:
1 The perspective-origin property is not supported in any browser.
2 Chrome and Safari support an alternative, the -webkit-perspecitve-origin property.

Syntax
perspective-origin: x-axis y-axis;

Property Value	Description
x-axis	Defining where the view is placed at the x-axis Possible values: left center right length % Default value: 50%
y-axis	Defining where the view is placed at the y-axis Possible values: top center bottom length % Default value: 50%

Example:
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>

```
<title>CSS3 Examples</title>
<style type="text/css">
#div1
{
position: relative;
height: 150px;
width: 150px;
margin: 50px;
padding:10px;
border: 1px solid black;
perspective:150;
perspective-origin: 10% 10%;
-webkit-perspective:150; /* Safari and Chrome */
-webkit-perspective-origin: 10% 10%; /* Safari and Chrome */
}
#div2
{
padding:50px;
position: absolute;
border: 1px solid black;
background-color: red;
transform: rotateX(45deg);
-webkit-transform: rotateX(45deg); /* Safari and Chrome */
}
</style>
</head>
<body>
<div id="div1">
  <div id="div2">HELLO</div>
</div>
</body>
</html>
```

CSS3 backface-visibility Property
The backface-visibility property defines whether or not an element should be visible when not facing the screen. This property is useful when an element is rotated, and you do not want to see its backside.

Syntax
backface-visibility: visible|hidden;

Value	**Description**
visible	The backside is visible

hidden The backside is not visible

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
position:relative;
height:60px;
width:60px;
background-color:red;
transform:rotateY(180deg);
-webkit-transform:rotateY(180deg); /* Chrome and Safari */
-moz-transform:rotateY(180deg); /* Firefox */
}
#div1
{
-webkit-backface-visibility:hidden;
-moz-backface-visibility:hidden;
}
#div2
{
-webkit-backface-visibility:visible;
-moz-backface-visibility:visible;
}
</style>
</head>
<body>
<p>This example shows two div elements, rotated 180 degrees, facing away from the user.</p>
<p>The first div element has the backface-visibility property set to "hidden", and should therefore be invisible.</p>
<div id="div1">DIV 1</div>
<div id="div2">DIV 2</div>
</body>
</html>
```

CSS3 Transitions

With CSS3, we can add an effect when changing from one style to another, without using Flash animations or JavaScripts.

A transition is such a property of CSS3, which is used to animate the object, without using flash or any other animation application. With this feature of CSS3 You can change the shape and size of your object with animated effects.

How does it work?
CSS3 transitions are effects that let an element gradually change from one style to another.

Transition Properties

Property	**Description**
transition	Shorthand property for setting all transition properties
transition-duration	Defines the length of time that a transition takes.Default 0
transition-delay	Specifies when the transition effect will start

CSS3 transition Property:
The transition property is a shorthand property for the four transition properties: transition-property, transition-duration, transition-timing-function, and transition-delay.

Syntax
transition: property duration timing-function delay;

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:100px;
background:blue;
transition:width 2s;
}
div:hover
{
width:300px;
}
</style>
</head>
```

```
<body>
<div></div>
<p>Hover over the div element above, to see the transition effect.</p>
</body>
</html>
```

CSS3 transition-duration Property

The transition-duration property specifies how many seconds (s) or milliseconds (ms) a transition effect takes to complete.

Syntax

transition-duration: time;

Value	**Description**
time	Specifies how many seconds or milliseconds a transition effect takes to complete.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:100px;
background:blue;
transition:width 2s;
transition-duration:5s;
}
div:hover
{
width:300px;
}
</style>
</head>
<body>
<div></div>
<p>Hover over the div element above, to see the transition effect.</p>
</body>
</html>
```

CSS3 transition-delay Property

The transition-delay property specifies when the transition effect will start. The transition-delay value is defined in seconds (s) or milliseconds (ms).

Syntax

transition-delay: time;

Value	**Description**
time	Specifies the number of seconds or milliseconds to wait before the transition effect will start

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:100px;
background:blue;
transition:width 2s;
transition-duration:5s;
transition-delay:2s;
}
div:hover
{
width:300px;
}
</style>
</head>
<body>
<div></div>
<p>Hover over the div element above, to see the transition effect.</p>
<p><b>Note:</b> The transition effect will wait 2 seconds before starting.</p>
</body>
</html>
```

CSS3 transition-property Property
The transition-property property specifies the name of the CSS property the transition effect is for.

Syntax
transition-property: none|all|property;

Value	**Description**
none	No property will get a transition effect
all	All properties will get a transition effect
property	Defines a comma separated list of CSS property names the transition effect.

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:100px;
background:red;
transition-property:width;
transition-duration: 2s;
}
div:hover
{
width:300px;
}
</style>
</head>
<body>
<div></div>
<p>Hover over the div element above, to see the transition effect.</p>
</body>
</html>
```

CSS3 transition-timing-function Property
The transition-timing-function property specifies the speed curve of the transition effect. This property allows a transition effect to change speed over its duration.

Syntax
transition-timing-function: linear|ease|ease-in|ease-out|ease-in-out|cubic-bezier(n,n,n,n);

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:100px;
background:red;
transition:width 2s;
transition-timing-function:linear;
}
div:hover
{
width:300px;
}
</style>
</head>
<body>
<div></div>
<p>Hover over the div element above, to see the transition effect.</p>
</body>
</html>
```

CSS3 Animations

With CSS3, we can create animations, which can replace animated images, Flash animations, and JavaScripts in many web pages. With this feature of CSS3 You can change the object into one style to another style in animated way.

@keyframes rule.
The @keyframes rule is where the animation is created. Specify a CSS style inside the @keyframes rule and the animation will gradually change from the current style to the new style.

CSS3 animation

When the animation is created in the @keyframe, bind it to a selector, otherwise the animation will have no effect. Bind the animation to a selector by specifying at least these two CSS3 animation properties:

1 Specify the name of the animation
2 Specify the duration of the animation

What are Animations in CSS3?
An animation is an effect that lets an element gradually change from one style to another.You can change as many styles you want, as many times you want. Specify when the change will happen in percent, or the keywords "from" and "to", which is the same as 0% and 100%. 0% is the beginning of the animation, 100% is when the animation is complete.

CSS3 Animation Properties

Property	Description
@keyframes	Specifies the animation
animation	A shorthand property for all the the animation properties
animation-name	Specifies the name of the @keyframes animation
animation-duration	Specifies how many seconds or milliseconds an animation takes to complete one cycle
animation-direction	Specifies whether or not the animation should play in reverse on alternate cycles
animation-delay	Specifies when the animation will start
animation-iteration-count	Specifies the number of times an animation should be played

CSS3 @keyframes Rule
With the @keyframes rule, you can create animations. The animation is created by gradually changing from one set of CSS styles to another. During the animation, you can change the set of CSS styles many times.

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
width:100px;
height:100px;
background:red;
position:relative;
animation:mymove 5s infinite;
}
@keyframes mymove
{
from {top:0px;}
to {top:200px;}
}
@-moz-keyframes mymove /* Firefox */
{
from {top:0px;}
to {top:200px;}
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

CSS3 animation Property:
The animation property is a shorthand property for six of the animation properties: animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, and animation-direction.

Syntax
animation: name duration timing-function delay iteration-count direction;

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
div
{
width:100px;
height:100px;
background:red;
position:relative;
animation:mymove 5s infinite;
}
```

```
@keyframes mymove
{
from {left:0px;}
to {left:200px;}
}
@-moz-keyframes mymove /*Firefox*/
{
from {left:0px;}
to {left:200px;}
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

CSS3 animation-duration Property:

The animation-duration property defines how many seconds or milliseconds an animation takes to complete one cycle.

Syntax

animation-duration: time;

Value	**Description**
time	Specifies the length an animation takes to finish. Default value is 0, meaning there will be no animation

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
div
{
width:100px;
height:100px;
background:red;
position:relative;
animation:mymove infinite;
animation-duration:1s;
}
@keyframes mymove
{
from {top:0px;}
to {top:200px;}
}
@-moz-keyframes mymove /* Firefox */
{
from {top:0px;}
to {top:200px;}
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

CSS3 animation-iteration-count Property
The animation-iteration-count property defines how many times an animation should be played.

Syntax
animation-iteration-count: value;

Value	**Description**
n	A number that defines how many times an animationshould be played
infinite	Specifies that the animation should be played infinite times(for ever)

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
div
{
width:100px;
height:100px;
background:red;
position:relative;
animation:mymove 5s;
animation-iteration-count:3;
}
@keyframes mymove
{
from {top:0px;}
to {top:200px;}
}
@-moz-keyframes mymove /* Firefox */
{
from {top:0px;}
to {top:200px;}
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

CSS3 animation-direction Property

The animation-direction property defines whether or not the animation should play in reverse on alternate cycles. If the animation-direction value is "alternate", the animation will be played as normal every odd time (1,3,5,etc..) and backwards every even time (2,4,6,etc...).

Syntax

animation-direction: value;

Value	**Description**
normal	Default value.The animation should be played as normal
alternate	The animation should play in reverse on alternate cycles

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
div
{
width:100px;
height:100px;
background:red;
position:relative;
animation:myfirst 5s infinite;
animation-direction:alternate;
}
@keyframes myfirst
{
0%   {background:red; left:0px; top:0px;}
25%  {background:yellow; left:200px; top:0px;}
50%  {background:blue; left:200px; top:200px;}
75%  {background:green; left:0px; top:200px;}
100% {background:red; left:0px; top:0px;}
}
@-moz-keyframes myfirst /* Firefox */
{
0%   {background:red; left:0px; top:0px;}
25%  {background:yellow; left:200px; top:0px;}
50%  {background:blue; left:200px; top:200px;}
75%  {background:green; left:0px; top:200px;}
100% {background:red; left:0px; top:0px;}
}
</style>
</head>
<body>
<div></div>
</body>
</html>
</body>
</html>
```

CSS3 Multiple Columns

With CSS3, you can create multiple columns for laying out text - like in newspapers!

1. column-count
2. column-gap
3. column-rule
4. column-fill
5. column-rule-color
6. column-rule-style
7. column-rule-width
8. column-span
9. column-width
10. columns

New Multiple Columns Properties

Property	Description
column-count	Specifies the no. of columns an element should bedivided.
column-fill	Specifies how to fill columns
column-gap	Specifies the gap between the columns
column-rule	A shorthand property for setting all the column-rule-*Propers.
column-rule-color	Specifies the color of the rule between columns
column-rule-style	Specifies the style of the rule between columns
column-rule-width	Specifies the width of the rule between columns
column-span	Specifies how many columns an element should span across
column-width	Specifies the width of the columns
columns	A shorthand property for setting column-width,column-count

CSS3 Create Multiple Columns

The column-count property specifies the number of columns an element should be divided into:

Syntax

column-count: number|auto;

Value	Description
number	The optimal number of columns into which the content of the element will be flowed
auto	The number of columns will be determined by other properties,

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
.newspaper
{
-moz-column-count:3;
column-count:3;
}
</style>
</head>
<body>
<div class="newspaper">
About RIAs
```

Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.

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

CSS3 Specify the Gap Between Columns
The column-gap property specifies the gap between the columns:

Syntax
column-gap: length|normal;

Value	**Description**
length	A specified length that will set the gap between the columns
normal	Specifies a normal gap between the columns.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
.newspaper
{
-moz-column-count:3;
column-count:3;
-moz-column-gap:40px;
column-gap:40px;
}
</style>
</head>
<body>
<div class="newspaper">
About RIAs
```

Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.

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

CSS3 column-rule Property

The column-rule property is a shorthand property for setting all the column-rule-* properties. The column-rule property sets the width, style, and color of the rule between columns.

Syntax

column-rule: column-rule-width column-rule-style column-rule-color;

Value	**Description**
column-rule-width	Sets the width of the rule between columns
column-rule-style	Sets the style of the rule between columns
column-rule-color	Sets the color of the rule between columns

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
.newspaper
{
-moz-column-count:3;
column-count:3;

-moz-column-gap:40px;
column-gap:40px;

-moz-column-rule:4px outset #ff00ff;
column-rule:4px outset #ff00ff;
}
```

```
</style>
</head>
<body>
<div class="newspaper">
About RIAs Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.
</div>
</body>
</html>
```

CSS3 column-fill Property

The column-fill property specifies how to fill columns, balanced or not.

Syntax

column-fill: balance|auto;

Value	Description
balance	Columns are balanced. Browsers should minimize the variation in column length
auto	Columns are filled sequentially, and they will have different lengths

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
column-fill:auto;
}
</style>
</head>
<body>
<div>
```

About RIAs Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.

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

CSS3 column-rule-color Property

The column-rule-color property specifies the color of the rule between columns.

Syntax

```
column-rule-color: color;
```

Value	**Description**
color	Specifies the color of the rule.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
.newspaper
{
column-count:3;
-moz-column-count:3;
column-gap:40px;
-moz-column-gap:40px;
column-rule-style:outset;
-moz-column-rule-style:outset;
column-rule-color:#ff0000;
-moz-column-rule-color:#ff0000;
}
</style>
</head>
<body>
<body>
```

```
<div class="newspaper">
About RIAs Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.
</div>
</body>
</html>
```

CSS3 column-rule-style Property
The column-rule-style property specifies the style of the rule between columns.

Syntax
column-rule-style: none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset;

Value	Description
none	Defines no rule
hidden	Defines a hidden rule
dotted	Defines a dotted rule
dashed	Defines a dashed rule
solid	Defines a solid rule
double	Defines a double rule
groove	Specifies a 3D grooved rule.
ridge	Specifies a 3D ridged rule.
inset	Specifies a 3D inset rule.
outset	Specifies a 3D outset rule.

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
.newspaper
{
-moz-column-count:3;
column-count:3;
-moz-column-gap:40px;
column-gap:40px;

-moz-column-rule-style:dotted;
column-rule-style:dotted;
}
</style>
</head>
<body>
<body>
<div class="newspaper">
```

About RIAs Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.

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

CSS3 column-rule-width Property
The column-rule-width property specifies the width of the rule between columns.

Syntax
column-rule-width: thin|medium|thick|length;

Value	**Description**
thin	Defines a thin rule
medium	Defines a medium rule
thick	Defines a thick rule
length	Specifies the width of the rule

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
.newspaper
```

```
{
-moz-column-count:4;
column-count:4;

-moz-column-gap:40px;
column-gap:40px;

-moz-column-rule-style:outset;
column-rule-style:outset;

-moz-column-rule-width:3px;
column-rule-width:3px;
}
</style>
</head>
<body>
<body>
<div class="newspaper">
About RIAs Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.
</div>
</body>
</html>
```

CSS3 column-span Property
The column-span property specifies how many columns an element should span across.

Syntax
column-span: 1|all;

Value	**Description**
1	The element should span across one column
all	The element should span across all columns

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
```

```
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
.newspaper
{
-webkit-column-count:3;
column-count:3;
}
h2
{
-webkit-column-span:all;
column-span:all;
}
</style>
</head>
<body>
<div class="newspaper">
<h2>RIAs Rich Internet applications (RIAs) in Web Environment</h2>
About RIAs Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.
</div>
</body>
</html>
```

CSS3 column-width Property
The column-width property specifies the width of the columns.

Syntax
column-width: auto|length;

Value	**Description**
auto	The column width will be determined by the browser
length	A length that specifies the width of the columns

Example:

```
<!DOCTYPE HTML>
```

```
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
.newspaper
{
-moz-column-width:100px; /* Firefox */
column-width:100px;
}
</style>
</head>
<body>
<div class="newspaper">
About RIAs Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.
</div>
</body>
</html>
```

CSS3 columns Property
The columns property is a shorthand property for setting column-width and column-count.

Syntax
columns: column-width column-count;

Value	**Description**
column-width	The width of the columns
column-count	The number of columns

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
```

```
.newspaper
{
columns:100px 3;
-moz-columns:100px 3;
}
</style>
</head>
<body>
<div class="newspaper">
About RIAs Rich Internet applications (RIAs) offer a rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops.
Silverlight is a new cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and Rich Interactive Applications(RIA) for the web. It runs in all popular browsers, including Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Opera. The plugin required to run Silverlight is very small in size hence gets installed very quickly.
</div>
</body>
</html>
```

CSS3 User Interface

A User Interface is a such property of CSS3, which gives an individual interface to users to implement their affords without editing the code section. As you can resize your div or can set your box size and e.t.c

There are mainly three types of User Interface that we use.
resize
box-sizing
outline-offset

New User-interface Properties

Property	Description
resize	Specifies whether or not an element is resizable by the user
box-sizing	Allows you to define certain elements to fit an area in a certain way
outline-offset	Offsets an outline, and draws it beyond the border edge

CSS3 Resizing

resize: is a such property of User Interface, by which you can resize your div layout on your browser. Three features of resize you can use
a) resize:both

b) resize:vertical
c) resize:horizontal

Syntax
resize: none|both|horizontal|vertical:

Value	**Description**
none	User cannot resize the element
both	User can adjust both the height and the width of a element.
horizontal	User can adjust the width of the element
vertical	User can adjust the height of the element

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
border:2px solid;
padding:10px 40px;
width:300px;
resize:both;
overflow:auto;
}
</style>
</head>
<body>
<div>The resize property specifies whether or not an element is resizable by the user.</div>
</body>
</html>
```

CSS3 box-sizing Property
The box-sizing property allows you to define certain elements to fit an area in a certain way.

Syntax
box-sizing: content-box|border-box|inherit:

Value	**Description**
content-box	This is the behavior of width and height as specified by CSS2.1
border-box	The specified width and height on this element determine

inherit	Specifies that the value of the box-sizing property should be inherited from the parent element

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div.container
{
width:30em;
border:1em solid red;
}
div.box
{
box-sizing:border-box;
-moz-box-sizing:border-box;
width:50%;
border:1em solid olive;
float:left;
}
</style>
</head>
<body>
<div class="container">
<div class="box">This div occupies the left half.</div>
<div class="box">This div occupies the right half.</div>
</div>
</body>
</html>
```

CSS3 outline-offset Property
The outline-offset property offsets an outline, and draws it beyond the border edge.
Outlines differ from borders in two ways:
1. Outlines do not take up space
2. Outlines may be non-rectangular

Syntax
outline-offset: length|inherit:

Value Description

length	The distance the outline is outset from the border edge
inherit	Specifies that the value of the outline-offset property should be inherited from the parent element

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
margin:20px;
width:150px;
padding:10px;
height:70px;
border:2px solid blue;
outline:2px solid red;
outline-offset:15px;
}
</style>
</head>
<body>
<div>This div has an outline border 15px outside the border edge.</div>
</body>
</html>
```

CSS3 appearance Property

The appearance property allows you to make an element look like a standard user interface element.

Syntax

appearance: normal|icon|window|button|menu|field;

Value	Description
normal	Render the element as normal
icon	Render the element as a small picture
window	Render the element as a viewport
button	Render the element as a button
menu	Render the element as a set of options for the user to choose from
field	Render the element as an input field

NOTE:
1. The appearance property is not supported in any of the major browsers.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
div
{
appearance:button;
-moz-appearance:button;
}
</style>
</head>
<body>
<div>Some text</div>
</body>
</html>
```

CSS3 icon Property

The icon property provides the author the ability to style an element with an iconic equivalent.

Syntax

icon: auto|URL|inherit;

Value	Description
auto	Uses a default generic icon provided by the browser
URL	Refers to one or more icons in a comma separated list
inherit	Specifies that the value of the icon property should be inherited from the parent element

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
img
```

```
{
content:icon;
icon:url(imgicon.png);
}
</style>
</head>
<body>
<img> Image if Supports..!!</img>
</body>
</html>
```

CSS3 nav-down Property:
The nav-down property specifies where to navigate when using the arrow-down navigation key.

Syntax
nav-down: auto|id|target-name|inherit;

Value	Description
auto	The browser determines which element to navigate to
id	Specifies the id of the element to navigate to
target-name	Specifies the target frame to navigate to
inherit	Specifies that the value of the nav-down property should be inherited from the parent element

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
button
{
position:absolute;
}
button#b1
{
top:20%;left:25%;
nav-index:1;
nav-right:#b2; nav-left:#b4;
nav-down:#b2; nav-up:#b4;
}
```

```
button#b2
{
top:40%;left:50%;
nav-index:2;
nav-right:#b3; nav-left:#b1;
nav-down:#b3; nav-up:#b1;
}
button#b3
{
top:70%;left:25%;
nav-index:3;
nav-right:#b4; nav-left:#b2;
nav-down:#b4; nav-up:#b2;
}
button#b4
{
top:40%;left:0%;
nav-index:4;
nav-right:#b1; nav-left:#b3;
nav-down:#b1; nav-up:#b3;
}
</style>
</head>
<body>
<button id="b1">Button 1</button>
<button id="b2">Button 2</button>
<button id="b3">Button 3</button>
<button id="b4">Button 4</button>
</body>
</html>
```

Selectors:
A selector is a chain of one or more simple selectors separated by combinators. Combinators are: white space, ">", and "+". White space may appear between a combinator and the simple selectors around it.

Types of Selectors:
1.[attribute^=value] Selector
2.[attribute$=value] Selector
3. [attribute*=value] Selector
4. first-of-type Selector
5. last-of-type Selector
6. only-of-type Selector

7. only-child Selector
8. nth-child() Selector
9. nth-last-child() Selector
10. nth-of-type() Selector
11. nth-last-of-type() Selector
12. last-child Selector
13. root Selector
14. empty Selector
15. target Selector

Advnced Selectors (CSS3)

1.[attribute^=value] Selector
The [attribute^=value] selector matches every element whose attribute value begins with a specified value.

Note: The [attribute^=value] selector is supported in all major browsers.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
[class^='Second']
{
color:red;font-family:tahoma;font-size:20px;
}
</style>
</head>
<body>
<div class='Hi_Second'>Welcome to CSS3 Selectors..! </div>
<p class='Second'>Welcome to CSS3 Selectors..! </p>
<div class='Second'>Welcome to CSS3 Selectors..! </div>
</body>
```

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
```

```
<title>CSS3 Examples</title>
<style type='text/css'>
[class^="test"]
{
background:#00ff99;
}
</style>
</head>
<body>
<div class="first_test">The first div element.</div>
<div class="test">The third div element.</div>
<p   class="test">This is some text in a paragraph.</p>
</body>
</html>
```

CSS3 [attribute$=value] Selector

The [attribute$=value] selector matches every element whose attribute value ends with a specified value.

Note: The [attribute$=value] selector is supported in all major browsers.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
[class$='Second']
{
color:red;font-family:tahoma;font-size:20px;
}
</style>
</head>
<body>
<div class='Hi_Second'>Welcome to CSS3 Selectors..! </div>
<p class='Second'>Welcome to CSS3 Selectors..! </p>
<div class='Second'>Welcome to CSS3 Selectors..! </div>
</body>
```

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
```

```
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
[class$="test"]
{
background:#00ff11;
}
</style>
</head>
<body>
<div class="first_test">The first div element.</div>
<div class="second">The second div element.</div>
<div class="test">The third div element.</div>
<p class="test">This is some text in a paragraph.</p>
</body>
</html>
```

CSS3 [attribute*=value] Selector
The [attribute*=value] selector matches every element whose attribute value containing a specified value.

Note: The [attribute*=value] selector is supported in all major browsers.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
div[class*='first']
{
color:red;font-family:tahoma;font-size:20px;
}
</style>
</head>
<body>
<div class='Hi_first'>Welcome to CSS3 Selectors..! </div>
<p class='Second'>Welcome to CSS3 Selectors..! </p>
<div class='first_hello'>Welcome to CSS3 Selectors..! </div>
</body>
</html>
```

CSS3:first-of-type Selector
The :first-of-type selector matches every element that is the first child, of a particular type, of its parent.

Note: The :first-of-type selector is supported in all major browsers, except IE8 and earlier.

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p:first-of-type
{
background:#00ffdd;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>
</body>
</html>
```

CSS3 :last-of-type Selector
The :last-of-type selector matches every element that is the last child, of a particular type, of its parent.

Note: The :last-of-type selector is supported in all major browsers, except IE8 and earlier.

Example:
```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p:last-of-type
```

```
{
background:#00ffdd;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>
</body>
</html>
```

CSS3 :only-of-type Selector
The :only-of-type selector matches every element that is the only child of its type, of its parent.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p:only-of-type
{
background:#ff0000;
}
</style>
</head>
<body>
<div><p>This is a paragraph.</p></div>
<div><p>This is a paragraph.</p>
<p>This is a paragraph.</p></div>
</body>
</html>
```

CSS3 :only-child Selector
The :only-child selector matches every element that is the only child of its parent.

Example:

```
<!DOCTYPE HTML>
```

```
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p:only-child
{
background:#ff0000;
}
</style>
</head>
<body>
<div><p>This is a paragraph.</p></div>
<div><span>This is a span.</span><p>This is a paragraph.</p></div>
</body>
</html>
```

CSS3 :nth-child() Selector
The :nth-child(n) selector matches every element that is the nth child, regardless of type, of its parent.

Note: The :nth-child() selector is supported in all major browsers, except IE8 and earlier.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p:nth-child(2)
{
background:#ffcc00;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>
</body>
```

```
</html>
```

CSS3 :nth-last-child() Selector

The :nth-last-child(n) selector matches every element that is the nth child, regardless of type, of its parent, counting from the last child. n can be a number, a keyword, or a formula.

Note: The :nth-last-child() selector is supported in all major browsers, except IE8 and earlier

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p:nth-last-child(2)
{
background:#ffcc00;
}
</style>
</head>
<body>
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>
</body>
</html>
```

CSS3 :nth-of-type() Selector

The :nth-of-type(n) selector matches every element that is the nth child, of a particular type, of its parent. n can be a number, a keyword, or a formula.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p:nth-of-type(2)
{
```

```
background:#ff0000;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>
</body>
</html>
```

CSS3 :nth-last-of-type() Selector
The :nth-last-of-type(n) selector matches every element that is the nth child, of a particular type, of its parent, counting from the last child. n can be a number, a keyword, or a formula.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p:nth-last-of-type(odd)
{
background:#ff0000;
}
p:nth-last-of-type(even)
{
background:#0000ff;
}
</style>
</head>
<body>
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
</body>
</html>
```

CSS3 :last-child Selector

The :last-child selector matches every element that is the last child of its parent.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type='text/css'>
p:last-child
{
background:#ffdd00;
}
</style>
</head>
<body>
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>
</body>
</html>
```

CSS3 :root Selector

The :root selector matches the document’s root element. In HTML, the root element is always the html element.

Note: It supports all major browsers.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
:root
{
background:#0000ff;
}
</style>
</head>
<body>
```

```
<h1>This is a heading</h1>
</body>
</html>
```

CSS3 :empty Selector
The :empty selector matches every element that has no children (including text nodes).

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
p:empty
{
width:100px;
height:20px;
background:#ff0000;
}
</style>
</head>
<body>
<p></p>
<p>A paragraph.</p>
<p>Another paragraph.</p>
</body>
</html>
```

CSS3 :target Selector
The :target selector can be used to style the current active target element.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
:target:before
{
content: url(water.gif);
}
```

```
</style>
</head>
<body>
<p><a href="#news1">Jump to New content 1</a></p>
<p><a href="#news2">Jump to New content 2</a></p>

<p>Click on the links above and the :target selector will add an image in front of the current
active HTML anchor.</p>

<p><a name="news1"></a><b>New content 1...</b></p>
<p><a name="news2"></a><b>New content 2...</b></p>
</body>
</html>
```

CSS3 :enabled Selector
The :enabled selector matches every enabled element (mostly used on form elements).

CSS3 :disabled Selector
The :disabled selector matches every disabled element (mostly used on form elements).

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
input[type="text"]:enabled
{
background:#00ffff;
}
input[type="text"]:disabled
{
background:#dddddd;
}
</style>
</head>
<body>
<form action="nit.html">
First name: <input type="text" value="Thomus" /><br/>
Last name: <input type="text" value="Affee" /><br/>
Country: <input type="text" disabled="disabled" value="INDIA" /><br/>
</form>
```

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

CSS3 :checked Selector
The :checked selector matches every checked input element (only for radio buttons or checkboxes)

Example

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
input:checked
{
background:#00ff00;
}
</style>
</head>
<body>
<form action="">
<input type="radio" checked="checked" value="male" name="gender"/> Male<br>
<input type="radio" value="female" name="gender" /> Female<br>
<input type="checkbox" checked="checked" value="Bike" /> I have a bike<br>
</body>
</html>
```

CSS3 :not Selector
The :not(selector) selector matches every element that is NOT the specified element/selector.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
p
{
color:#000000;
}
:not(p)
{
color:#ff0000;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<div>This is some text in a div element.</div>
<a href="http://www.nareshit.com"
target="_blank">Link to NareshIT!</a>
</body>
</html>
```

CSS3 ::selection Selector
The ::selection selector matches the portion of an element that is selected by a user. Only a few CSS properties can be applied to ::selection: color, background, cursor, and outline.

Example:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset=utf-8>
<title>CSS3 Examples</title>
<style type="text/css">
::selection
{
color:#ff0000;
}
::-moz-selection
{
color:#ff0000;
}
</style>
</head>
<body>
<h1>Select Text and Observe</h1>
<p>This is a paragraph.</p>
<div>This is some text in a div element.</div>
<a href="http://www.nareshit.com"
```

```
target="_blank">Link to NareshIT!</a>
</body>
</html>
```