

2023

WebDesign Course book



Index

Introduction to HTML.....	6
HTML-Basic Tags.....	9
HTML-Element	13
HTML-Comments	Error! Bookmark not defined.
HTML Frames	15
HTML Tables.....	17
HTML Forms and Input	22
HTML Fonts	24
HTML –Marquee	25
HTML-Multimedia.....	21
 Cascading Style Sheet	 35
CSS-Background	44
Css3 notes	62
CSS3 Animation:	62
CSS3 Borders:	62
CSS3 Fonts:	63
CSS3 Multiple Columns:.....	63
CSS3 Text Effects:	63
CSS3 Transition Effects:	64
transition-property	64
transition-duration	65
transition-timing-function	65
transition-delay	65
CSS3 User Interface:	65
CSS3 2D Transform:	66
translate() method.....	66
Bootstrap-notes	67
What is Bootstrap?.....	67
Where to Get Bootstrap?	67

What is Bootstrap Grid System?	68
Mobile First Strategy	68
Working of Bootstrap Grid System:	68
Media Queries	69
Grid options	70
Basic Grid Structure.....	71
Basic Table:.....	72
Optional Table Classes.....	73
Striped Table	73
Bordered Table	74
Bootstrap Form:	75
Vertical or Basic Form	75
Inline Form.....	76
Horizontal Form	76
Bootstrap-button:	77
Button Size	78
Javascript	78
DataTypes and Variable Declaration	79
WritingJavaScript	80
JavaScriptDialogBoxes	80
ExternalJavaScript	82
Switch	83
TernaryOperator(?:).....	84
LoopingControlStatements	84
While.....	84
For	84
Working With Arrays:	84
TwoDimesionalArrays	85
Validations	89
Cookies	93
DOM(DocumentObjectModel)"Window"Object	96
DOM(DocumentObjectModel)"Window"ObjectProperties	96
DOM(DocumentObjectModel)"Window"ObjectMethods	98

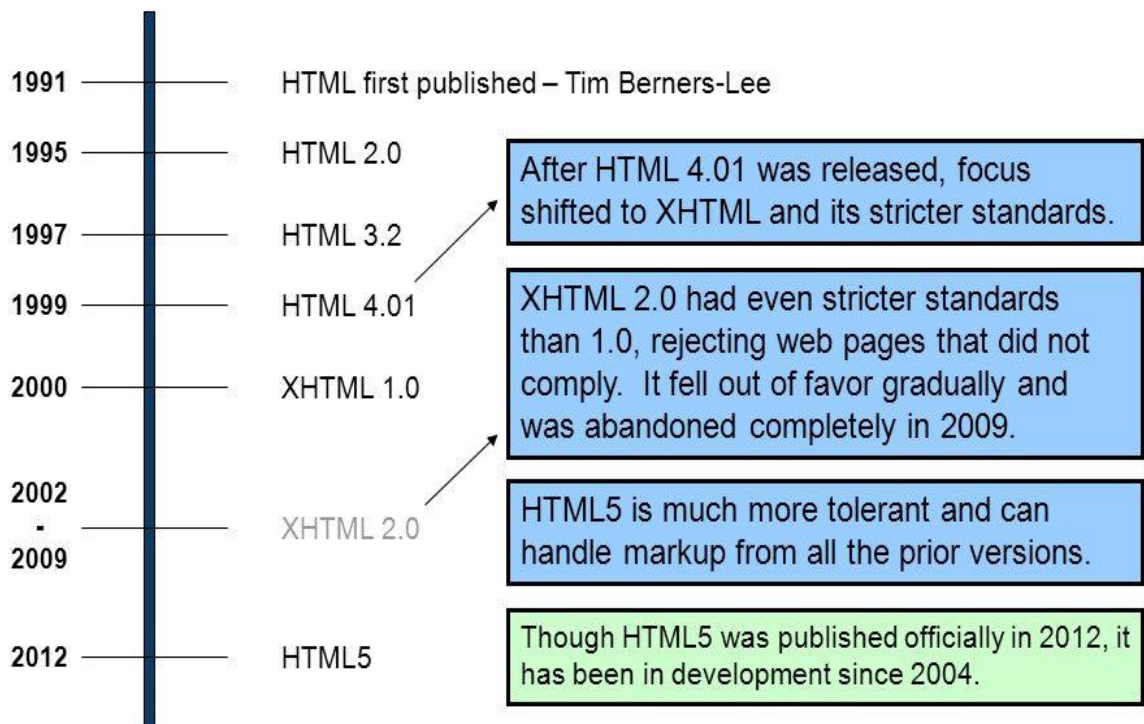
DOM(DocumentObjectModel)"Document"Object	99
DOM(DocumentObjectModel)"Document"ObjectProperties	99
DOM(DocumentObjectModel)"Document"ObjectMethods	100

HTML

What is an HTML File?

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

History of HTML



Difference between HTML and HTML5

HTML	HTML5
DOCTYPE declaration too lengthy and refers to an external resource.	DOCTYPE declaration is simple and in one line , for example : <!DOCTYPE html>.
It's not mobile friendly.	It's mobile friendly.
No Audio and Video support in HTML.	Audio/video elements can be integrated directly onto a web page.
It does not support all major web browsers.	It is supported by all major web browsers.
Vector Graphics is possible when in conjunction with flash, Silverlight, or similar third-party plugins.	Scalable Vector Graphics (SVG) is an integral parts of the HTML5 language specification.
It does not allow JavaScript to run a in browser.	I allow JavaScript to run in background.

Basic HTML Structure

```

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

```

Basic Example

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
  This is my first homepage. <b>This text is bold</b>
</body>
</html>
```

Tag	Description
<!DOCTYPE...>	This tag defines the document type and HTML version.
<html>	This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.
<head>	This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.
<title>	The <title> tag is used inside the <head> tag to mention the document title.
<body>	This tag represents the document's body which keeps other HTML tags like <h1>, <div>, <p> etc.

HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, and can make the font bigger or smaller, and so on. For example, take the following line of content:

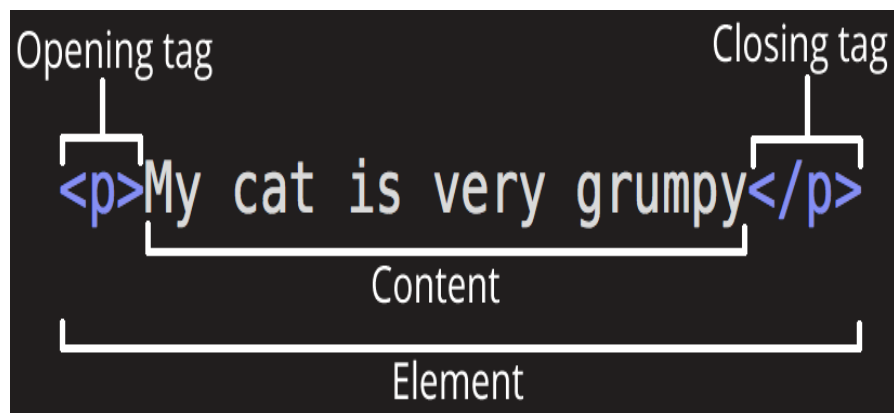
My cat is very grumpy

If we wanted the line to stand by itself, we could specify that it is a paragraph by enclosing it in paragraph tags:

<p> My cat is very grumpy </p>

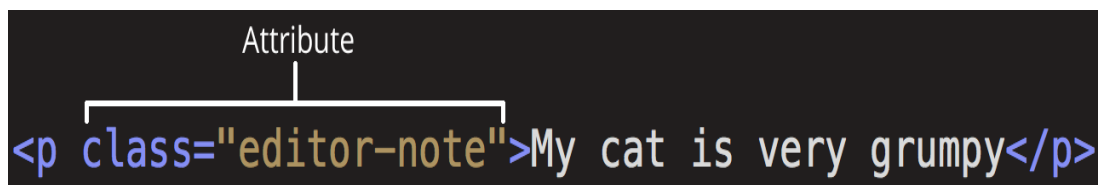
Anatomy of an HTML element

Let's explore this paragraph element a bit further.



The main parts of our element are as follows:

1. **The opening tag:** This consists of the name of the element (in this case, p), wrapped in opening and closing **angle brackets**. This states where the element begins or starts to take effect — in this case where the paragraph begins.
 2. **The closing tag:** This is the same as the opening tag, except that it includes a *forward slash* before the element name. This states where the element ends — in this case where the paragraph ends. Failing to add a closing tag is one of the standard beginner errors and can lead to strange results.
 3. **The content:** This is the content of the element, which in this case, is just text.
 4. **The element:** The opening tag, the closing tag, and the content together comprise the element.
- Elements can also have attributes that look like the following:



```
<p class="editor-note">My cat is very grumpy</p>
```

HTML-Basic Tags

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

1.Headings:

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

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
<h4>This is a heading</h4>
<h5>This is a heading</h5>
<h6>This is a heading</h6>
```

HTML automatically adds an extra blank line before and after a heading. Paragraphs

2.Paragraphs:

are defined with the <p> tag.

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

HTML automatically adds an extra blank line before and after a paragraph.

3. Line Breaks:

The
 tag is used when you want to end a line, but don't want to start a new paragraph. The
 tag forces a line break wherever you place it.

```
<p>This<br> is a para<br>graph with line breaks</p>
```

The
 tag is an empty tag. It has no closing tag.

4. Horizontal Rule:

The <hr> tag is used when you want to Horizontal a line

```
<p> Welcome to enosis learning</p> <hr>
```

```
<h1> enosis learning is a great learning place</h1>
```

5. Comments in HTML:

The comment tag is used to insert a comment in the HTML source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.

```
<!-- This is a comment -->
```

Note that you need an exclamation point after the opening bracket, but not before the closing bracket.

Comments in CSS:

The comment tag is used to insert a comment in the CSS head of source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.

```
/* This is a comment */
```

Basic HTML Tags:

Tag	Description
<html>	Defines an HTML document
<body>	Defines the document's body
<h1> to <h6>	Defines header 1 to header 6
<p>	Defines a paragraph

	Inserts a single line break
<hr>	Defines a horizontal rule
<!-->	Defines a comment

HTML Text Formatting:

HTML defines a lot of elements for formatting output, like bold or italic text.

Text Formatting Tags:

Tag	Description
	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
	Defines deleted text
<s>	Deprecated. Use instead
<strike>	Deprecated. Use instead

HTML-Element

An **HTML element** is defined by a starting tag. If the element contains other content, it ends with a closing tag, where the element name is preceded by a forward slash as shown below with few tags

So here `<p>...</p>` is an HTML element, `<h1>...</h1>` is another HTML element. There are some HTML elements which don't need to be closed, such as `<img.../>`, `<hr />` and `
` elements. These are known as **void elements**.

HTML documents consists of a tree of these elements and they specify how HTML documents should be built, and what kind of content should be placed in what part of an HTML document.

Anchor Tag and the href Attribute:

HTML uses the `<a>` (anchor) tag to create a link to another document.

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

The syntax of creating an anchor:

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

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

This anchor defines a link to enosis learning:

```
<a href="http://www.enosislearning.com/">EnosisLearning</a>
```

The Target Attribute

With the target attribute, you can define where the linked document will be opened.

The line below will open the document in a new browser window:

```
<a href="http://www.enosislearning.com/" target="_blank">Enosis Learning!</a>The  
Anchor Tag and the Name Attribute
```

The name attribute is used to create a named anchor. When using named anchors we can create links that can jump directly into a specific section on a page, instead of letting the user scroll around to find what he/she is looking for.

Below is the syntax of a named anchor:

```
<a name="label">Text to be displayed</a>
```

The name attribute is used to create a named anchor. The name of the anchor can be any text you care to use.

The line below defines a named anchor:

```
<a name="tips">Useful Tips Section</a>
```

You should notice that a named anchor is not displayed in a special way.

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

```
<a href="http://www.enosislearning.com/html_links.asp#tips">
Jump to the Useful Tips Section</a>
```

A hyperlink to the Useful Tips Section from WITHIN the file "html_links.asp" will look like this:

```
<a href="#tips">Jump to the Useful Tips Section</a>
```

A hyperlink or anchor tag can be used as an image also and it will look like this:

```
<a href="#anypage"></a>
```

Link Tags:

Tag	Description
<a>	Defines an anchor
<href>	Source of file

HTML-Images

Images are very important to beautify as well as to depict many complex concepts in simple way on your web page. This tutorial will take you through simple steps to use images in your web pages.

Insert Image

You can insert any image in your web page by using **** tag. Following is the simple syntax to use this tag.

```
<imgsrc="Image URL" ... attributes-list/>
```

The **** tag is an empty tag, which means that, it can contain only list of attributes and it has no closing tag

The **alt** attribute is a mandatory attribute which specifies an alternate text for an image, if the image cannot be displayed.

Set Image Location

Usually we keep all the images in a separate directory. So let's keep HTML file test.htm in our home directory and create a subdirectory **images** inside the home directory where we will keep our image test.png.

Set Image Width/Height

You can set image width and height based on your requirement using **width** and **height** attributes. You can specify width and height of the image in terms of either pixels or percentage of its actual size.

Example

```
<html>
<head>
<title>Set Image Width and Height</title> HTML
48 </head>
<body>
<p>Setting image width and height</p>
<imgsrc="test.png" alt="Test Image" width="150" height="100"/>
</body>
</html>
```

Output:

This will produce the following result:

Setting image width and height

HTML Frames

With frames, you can display more than one Web page in the same browser window. Examples

Vertical frameset:

This example demonstrates how to make a vertical frameset with three different documents.

Horizontal frameset:

This example demonstrates how to make a horizontal frameset with three different documents.

How to use the <noframes> tag

This example demonstrates how to use the <noframes> tag.

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

- The web developer must keep track of more HTML documents
- It is difficult to print the entire page

The Frameset Tag

- The <frameset> tag defines how to divide the window into frames
- Each frameset defines a set of rows or columns
- The values of the rows/columns indicate the amount of screen area each row/column will occupy

The Frame Tag

- The <frame> tag defines what HTML document to put into each frame

In the example below we have a frameset with two columns. The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The HTML document "frame_a.htm" is put into the first column, and the HTML document "frame_b.htm" is put into the second column:

```
<frameset cols="25%,75%">
<frame src="frame_a.htm">
<frame src="frame_b.htm">
</frameset>
```

Basic Notes - Useful Tips

If a frame has visible borders, the user can resize it by dragging the border. To prevent a user from doing this, you can add noresize="noresize" to the <frame> tag.

Add the <noframes> tag for browsers that do not support frames.

Frame Tags:

Tag	Description
<frameset>	Defines a set of frames
<frame>	Defines a sub window (a frame)
<noframes>	Defines a noframe section for browsers that do not handle frames

```
<!DOCTYPEhtml>
<htmlxmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
</head>
```



```
<!-- <frameset rows="16%,84%">
<frame src="1.html" name="top"/>
<frame src="2.html" name="bottom"/>
</frameset> -->
```

```
<framesetrows="16%,84%">
<framesetcols="50%,50%">
<framesrc="1.html"name="t1">
<framesrc="2.html"name="tr">
</frameset>
<framesrc="1.html"name="bottom">
</frameset>
</html>
```

HTML Tables

With HTML you can create tables.

Tables:

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

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

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Tables and the Border Attribute:

If you do not specify a border attribute the table will be displayed without any borders. Sometimes this can be useful, but most of the time, you want the borders to show. To display a table with borders, you will have to use the border attribute:

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

Headings in a Table:

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

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

How it looks in a browser:

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Empty Cells in a Table

Table cells with no content are not displayed very well in most browsers.

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

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Note that the borders around the empty table cell are missing.

To avoid this, add a non-breaking space () to empty data cells, to make the borders visible:

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>&nbsp;</td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Table Tags:

Tag	Description
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell

HTML Lists

HTML supports ordered, unordered and definition lists.

Unordered Lists

An unordered list is a list of items. The list items are marked with bullets (typically small black circles).

An unordered list starts with the `` tag. Each list item starts with the `` tag.

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

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

Ordered Lists

An ordered list is also a list of items. The list items are marked with numbers.

An ordered list starts with the `` tag. Each list item starts with the `` tag.

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

Here is how it looks in a browser:

1. Coffee
2. Milk

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

Definition Lists:

A definition list is not a list of items. This is a list of terms and explanation of the terms.

A definition list starts with the <dl> tag.

Each definition-list term starts with the <dt> tag.

Each definition-list definition starts with the <dd> tag.

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

Here is how it looks in a browser:

```
Coffee
    Black hot drink
Milk
    White cold drink
```

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

List Tags:

Tag	Description
	Defines an ordered list
	Defines an unordered list
	Defines a list item
<dl>	Defines a definition list
<dt>	Defines a definition term
<dd>	Defines a definition description

HTML Forms and Input

HTML Forms are used to select different kinds of user input.

Forms: A form is an area that can contain form elements.

Form elements are elements that allow the user to enter information (like text fields, text area fields, drop-down menus, radio buttons, checkboxes, etc.) in a form.

A form is defined with the <form> tag.

```
<form>
<input>
<input>
</form>
```

Input : The most used form tag is the <input> tag. The type of input is specified with the type attribute. The most commonly used input types are explained below.

Text Fields: Text fields are used when you want the user to type letters, numbers, etc. in a form.

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

How it looks in a browser:

First name:

Last name:

Note that the form itself is not visible. Also note that in most browsers, the width of the text field is 20 characters by default.

Radio Buttons: Radio Buttons are used when you want the user to select one of a limited number of choices.

```
<form>
```

```
<input type="radio" name="sex" value="male"> Male
<br>
<input type="radio" name="sex" value="female"> Female
</form>
```

Checkboxes: Checkboxes are used when you want the user to select one or more options of a limited number of choices.

```
<form>
<input type="checkbox" name="bike">
I have a bike
<br>
<input type="checkbox" name="car">
I have a car
</form>
```

Text area: The `<textarea>` tag defines a multi-line text input control. The size of a text area is specified by the `<cols>` and `<rows>` attributes (or with CSS).

```
<form>

<textarea id="w3review" name="w3review" rows="4" cols="50">At w3schools.com you
will learn how to make a website. They offer free tutorials in all web development
technologies.</textarea>

</form>
```

The Form's Action Attribute and the Submit Button

When the user clicks on the "Submit" button, the content of the form is sent to another file. The form's action attribute defines the name of the file to send the content to. The file defined in the action attribute usually does something with the received input.

```
<form name="input" action="html_form_action.asp" method="get">
Username:
<input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

How it looks in a browser:

Username:

If you type some characters in the text field above, and click the "Submit" button, you will send your input to a page called "html_form_action.asp". That page will show you the received input.

Form Tags:

Tag	Description
<form>	Defines a form for user input
<input>	Defines an input field
<textarea>	Defines a text-area (a multi-line text input control)
<label>	Defines a label to a control
<fieldset>	Defines a fieldset
<legend>	Defines a caption for a fieldset
<select>	Defines a selectable list (a drop-down box)
<optgroup>	Defines an option group
<option>	Defines an option in the drop-down box
<button>	Defines a push button

HTML Fonts

The tag in HTML is deprecated. It is supposed to be removed in a future version of HTML. Even if a lot of people are using it, you should try to avoid it, and use styles instead.

The HTML Tag

With HTML code like this, you can specify both the size and the type of the browser output :

```
<p>
<font size="2" face="Verdana">
This is a paragraph.
</font>
</p>
<p>
<font size="3" face="Times">
```


This is another paragraph.

</p>

Font Attributes :

Attribute	Example	Purpose
size="number"	size="2"	Defines the font size
size="+number"	size="+1"	Increases the font size
size="-number"	size="-1"	Decreases the font size
face="face-name"	face="Times"	Defines the font-name
color="color-value"	color="#eeff00"	Defines the font color
color="color-name"	color="red"	Defines the font color

The Tag Should NOT be Used

The tag is deprecated in the latest versions of HTML (HTML 4 and XHTML).

HTML –Marquee

An HTML marquee is a scrolling piece of text displayed either horizontally across or vertically down your webpage depending on the settings. This is created by using HTML <marquees> tag.

Note: The HTML <marquee> tag may not be supported by various browsers so it is not recommended to rely on this tag, instead you can use JavaScript and CSS to create such effects.

Syntax

A simple syntax to use HTML <marquee> tag is as follows:

<marquee attribute_name="attribute_value"....more attributes>

One or more lines or text message or image

</marquee>

HTML 5

HTML5 is a next version of HTML. Here, you will get some brand new features which will make HTML much easier. These new introducing features make your website layout clearer to both website designers and users. There are some elements like <header>, <footer>, <nav> and <article> that define the layout of a website.

Why use HTML5

It is enriched with advance features which makes it easy and interactive for designer/developer and users.

It allows you to play a video and audio file.

It allows you to draw on a canvas.

It facilitate you to design better forms and build web applications that work offline.

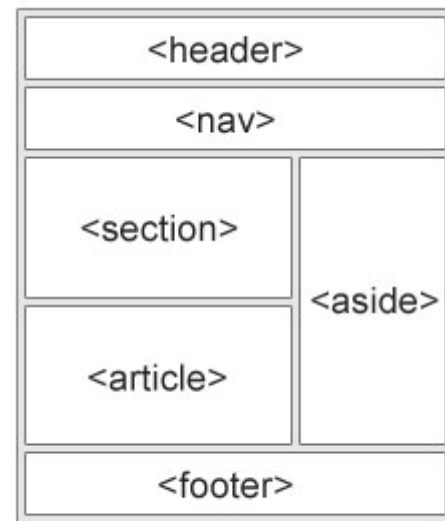
It provides you advance features for that you would normally have to write JavaScript to do.

HTML 5 Example

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<title>Page Title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph. </p>
</body>
</html>
```

HTML5 Layout

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



HTML5 –Multimedia

Multimedia comes in many different formats. It can be almost anything you can hear or see.

Examples: Images, music, sound, videos, records, films, animations, and more. Web pages often contain multimedia elements of different types and formats.

- ☐ The HTML5 `<video>` element specifies a standard way to embed a video in a web page.
- ☐ `<video width="320" height="240" controls>`
`<source src="movie.mp4" type="video/mp4">`
`</video>`
- ☐ To play an audio file in HTML, use the `<audio>` element:
- ☐ `<audio controls>`
`<source src="horse.mp3" type="audio/mp3"></audio>`

HTML5 CANVAS

What is HTML Canvas?

- The HTML CANVAS element is used to draw graphics, on the fly, via scripting (usually JavaScript).
- The CANVAS element is only a container for graphics. You must use a script to actually draw the graphics.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Example:

- Inserting CANVAS tag
- Line Using Canvas
- ARC Using Canvas
- Rectangle Using canvas
- Stroke Using canvas
- Cap Stroke Using canvas
- Inside a rectangle
- Inside a circle

1) Inserting CANVAS tag

2) <html>

```
<html lang="en"><head>
<meta charset="UTF-8">
<title>HTML5 Canvas</title>
<script type="text/javascript">
    window.onload = function(){
        var canvas = document.getElementById("myCanvas");
        // draw stuff here
    };
</script>
<style>
canvas{border:1px solid #000}
</style>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body></html>
```

3) Line Using Canvas

```
<html lang="en"><head>
<meta charset="UTF-8">
<title>Drawing a Line on Canvas</title>
<style type="text/css">
canvas{border: 1px solid #000;}
</style>
<script type="text/javascript">
    window.onload = function(){
        var canvas = document.getElementById("myCanvas");
```

```

        var context = canvas.getContext("2d");
        context.moveTo(20, 20);
        context.lineTo(100, 150);
        context.lineTo(300, 300);
        context.lineTo(75, 100);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="700" height="400"></canvas>
</body></html>

```

4) ARC Using Canvas

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Drawing an Arc on Canvas</title>
<style type="text/css">
canvas{border: 1px solid #000;}
</style>
<script type="text/javascript">
    window.onload = function(){
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        //context.arc(centerX, centerY, radius, startingAngle, endingAngle, counterclockwise);
        context.arc(150, 150, 50, 0 * Math.PI, 0.5 * Math.PI, true);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="500" height="500"></canvas>
</body></html>

```

5) Rectangle Using canvas

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Drawing a Rectangle on Canvas</title>
<style type="text/css">
canvas{border: 1px solid #000;}
</style>
<script type="text/javascript">
    window.onload = function(){
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.lineWidth = 2;
        context.strokeStyle = "red";
        //context.rect(x, y, width, height);
        context.rect(50, 50, 200, 200);
        context.stroke();
    };

```

```

    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="300"></canvas>
</body></html>

```

6) Stroke Using canvas

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Example of Setting Stroke Color and Width</title>
<style type="text/css">
canvas{border: 1px solid #000; }
</style>
<script type="text/javascript">
    window.onload = function(){
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.lineWidth = 5;
        context.strokeStyle = "blue";
        context.moveTo(50, 150);
        context.lineTo(250, 50);
        context.stroke();
        context.lineCap = "round";
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body></html>

```

7) Cap Stroke Using canvas

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Example of Setting Stroke Cap Style</title>
<style type="text/css">
canvas{border: 1px solid #000;}
</style>
<script type="text/javascript">
    window.onload = function(){
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.lineWidth = 10;
        context.strokeStyle = "orange";
        //butt, round, and square.
        context.lineCap = "butt";
        context.arc(150, 150, 80, 1.2 * Math.PI, 1.8 * Math.PI, false);
        context.stroke();
    };
</script>

```

```

</head>
<body>
  <canvas id="myCanvas" width="300" height="200"></canvas>
</body></html>

```

8) Inside a rectangle

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Example of Filling Color inside a Rectangle</title>
<style type="text/css">
canvas{border: 1px solid #000;}
</style>
<script type="text/javascript">
  window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    context.rect(50, 50, 200, 100);
    context.fillStyle = "#999";
    //It is recommended to use the fill() method before the stroke() method in order to render the stroke
    correctly.
    context.fill();
    context.lineWidth = 5;
    context.strokeStyle = "blue";
    context.stroke();
  };
</script>
</head>
<body>
  <canvas id="myCanvas" width="300" height="200"></canvas>
</body></html>

```

9) Inside a circle

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Example of Filling Color inside a Circle</title>
<style type="text/css">
canvas{border: 1px solid #000;}
</style>
<script type="text/javascript">
  window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    context.arc(150, 100, 70, 0, 2 * Math.PI, false);
    context.fillStyle = "#FB8B89";
    //It is recommended to use the fill() method before the stroke() method in order to render the stroke
    correctly.
    context.fill();
    context.lineWidth = 5;
    context.strokeStyle = "black";
    context.stroke();
  };

```

```

    };
  </script>
</head>
<body>
  <canvas id="myCanvas" width="300" height="200"></canvas>
</body></html>

```

HTML5 SVG

What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define vector-based graphics for the Web
- SVG defines the graphics in XML format
- SVG graphics do NOT lose any quality if they are zoomed or resized
- Every element and every attribute in SVG files can be animated

SVG Example

- Start With svg
- Line Using SVG
- Rectangle Using SVG
- Circle Using SVG
- Text Using SVG
- Transform Using SVG

1) Start With svg

```

<html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Embedding SVG Into HTML Pages</title>
<style type="text/css">
svg{border:1px solid #000}
</style>
</head>
<body>
  <svg width="600" height="200">
    <text x="10" y="20" style="font-size:14px; color:#ccc">
      Your browser support SVG.
    </text>
    Sorry, your browser does not support SVG.
  </svg>
</body>
</html>

```

2) Line Using SVG

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Create a Line with HTML5 SVG</title>

```



```

<style type="text/css">
  svg {border: 1px solid black; }
</style>
</head>
<body>
  <svg width="300" height="200">
    <line x1="10" y1="10" x2="150" y2="100" style="stroke:#999; stroke-width:1;"></line>
  </svg>
</body></html>

```

3) Rectangle Using SVG

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Create a Rectangle with HTML5 SVG</title>
<style type="text/css">
  svg {border: 1px solid black;}
</style>
</head>
<body>
  <svg width="300" height="200">
    <rect x="50" y="50" width="200" height="100" style="fill:red; stroke:yellow; stroke-
width:5;"></rect>
  </svg>
</body></html>

```

4) Circle Using SVG

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Create a Circle with HTML5 SVG</title>
<style type="text/css">
  svg {border: 1px solid black;}
</style>
</head>
<body>
  <svg width="300" height="200">
    <circle cx="150" cy="100" r="100" style="fill:green; stroke:red; stroke-width:1;"></circle>
  </svg>
</body></html>

```

5) Text Using SVG

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Render Text with HTML5 SVG</title>
<style type="text/css">
  svg {border: 1px solid black; }
</style>
</head>
<body>
  <svg width="300" height="200">
    <text x="20" y="30" style="fill:purple; font-size:22px; transform:rotate(10deg);">
      Welcome to Our Website!

```

```

</text>
<text x="20" y="30" dx="0" dy="20" style="fill:navy; font-size:14px; transform:rotate(45deg);">
  Here you will find lots of useful information.
</text>
</svg>

```

```

</body></html>

```

6) Transform Using SVG

```

<html lang="en"><head>
<meta charset="UTF-8">
<title>Rotate and Render Text with HTML5 SVG</title>
<style type="text/css">
  svg {border: 1px solid black;}
</style>
</head>
<body>
  <svg width="300" height="200">
    <text x="30" y="15" style="fill:purple; font-size:22px; transform:rotate(45deg);">
      <tspan style="fill:purple; font-size:22px;">
        Welcome to Our Website!
      </tspan>
      <tspan dx="-230" dy="20" style="fill:navy; font-size:14px;">
        Here you will find lots of useful information.
      </tspan>
    </text>
  </svg>
</body></html>

```

Difference between SVG and CANVAS

SVG	Canvas
Vector based (composed of shapes)	Raster based (composed of pixel)
Multiple graphical elements, which become the part of the DOM	Single HTML element similar to <code></code> in behavior
Modified through script and CSS	Modified through script only
Give better performance with smaller number of objects or larger surface, or both	Give better performance with smaller surface or larger number of objects, or both
Better scalability — can be printed with high quality at any resolution	Poor scalability — not suitable for printing on higher resolution

CSS(Cascading Style Sheets)

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.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, as well as a variety of other effects.

CSS is easy to learn and understand but it provides a powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

History of CSS :

- **Css1 (1996)** : Enabled users to set font size; align text center , left , or right; set body margins ; and apply background and foreground colors to page elements.
- **Css2 (1998)**: included features such as design style for different output devices such as print media and aural devices, and controlling the appearance and behavior of browse features.
- **Css3 (2005)**: includes text effects such as drop shadow and web fonts, semitransparent colors, box outlines, and rotating page elements.

Advantages of CSS:

- **CSS saves time** - You can write CSS once and then reuse the same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many web pages as you want.
- **Pages load faster** - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.
- **Easy maintenance** - To make a global change, simply change the style, and all the elements in all the web pages will be updated automatically.
- **Superior styles to HTML** - CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cellphones or for printing.
- **Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible with future browsers.

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.

You can put CSS Style Rule Syntax as follows:

```
selector { property: value }
```

Example: You can define a table border as follows:

```
table{ border :1px solid #C00; }
```

Here table is a selector and border is a property and the given value *1px solid #C00* is the value of that property.

You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one. Three ways to write css:

CSS information can be provided from various sources. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported.

- ❖ **Inline styles**, inside the HTML document, style information on a single element, specified using the style attribute
- ❖ **Embedded or internal style**, blocks of CSS information inside the <head> element of HTML itself
- ❖ **External style sheets**, i.e., a separate CSS file referenced from the document

Example : The following example demonstrates the inline style

```
<!doctype html>
<html>
<body>
<p style="font-family:arial;color:blue;font-size:medium">
  This is a paragraph formatted with inline css style.
</p>
</body>
</html>
```

Example : The following example demonstrates the internal style

```
<!doctype html>
<html>
<head>
<style>
P
{
  Font-family:arial;
  Font-size:medium;
  Color:blue;
  Text-align:justify;
}
</style>
</head>
<body>
<p>This is a paragraph formatted with internal css style.
</p>
</body>
</html>
```

Example : The following example demonstrates the external style

Create a new document in notepad, define the following css styles within it and save it with the name MyStyle.css

```
P
{
  Font-family:arial;
  Font-size:medium;
  Color:blue;
  Text-align:justify;
}
H1
{
  Font-family:arial black;
  Background-color:blue;
  Color:white;
}
```

Create another new document in notepad and create a html file with the following html markup. To link an external css file, use <link>element of html within the <head> element.

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="MyStyle.css">
</head>
<body>
<h1> Heading </h1>
<p>This is a paragraph formatted with internal css style. </p>
</body>
</html>
```

Grouping:

When several selectors share the same declarations, they may be grouped into a comma separated list.

In this example, we condense three rules with identical declarations into one. Thus,

h1 {font-family: sans-serif } h2 { font-family: sans-serif } h3 { font-family: sans-serif } is equivalent to:

h1, h2, h3 { font-family: sans-serif }

Universal selector:

The universal selector, written "*", matches the name of any element type. It matches any single element in the document tree. If the universal selector is not the only component of a simple selector, the "*" may be omitted. For example:

**[lang=fr] and [lang=fr] are equivalent.*

**.warning and .warning are equivalent.*

**#myid and #myid are equivalent.*

Type selectors:

A *type selector* matches the name of a document language element type. A type selector matches every instance of the element type in the document tree.

The following rule matches all H1 elements in the document tree:

h1 { font-family: sans-serif }

Descendant selectors:

At times, authors may want selectors to match an element that is the descendant of another element in the document tree (e.g., "Match those EM elements that are contained by an H1 element"). Descendant selectors express such a relationship in a pattern. A descendant selector is made up of two or more selectors separated by white space. A descendant selector of the form "A B" matches when an element B is an arbitrary descendant of some ancestor element A. For example, consider the following rules:

h1 { color: red } em { color: red }

Although the intention of these rules is to add emphasis to text by changing its color, the effect will be lost in a case such as:

<H1>This headline is very important</H1>

We address this case by supplementing the previous rules with a rule that sets the text color to blue whenever an EM occurs anywhere within an H1:

h1 { color: red } em { color: red } h1 em

{ color: blue }

The third rule will match the EM in the following fragment:

```
<H1>This<SPAN class="myclass">headline is<EM>very</EM>
important</SPAN></H1>
```

The following selector: *div * p*

matches a P element that is a grandchild or later descendant of a DIV element. Note the white space on either side of the "*" is not part of the universal selector; the white space is a combinator indicating that the DIV must be the ancestor of some element, and that that element must be an ancestor of the P.

The selector in the following rule, which combines descendant and attribute selectors, matches any element that has the "href" attribute set and is inside a P that is itself inside a DIV:

```
div p *[href]
```

Child selectors:

A *child selector* matches when an element is the child of some element. A child selector is made up of two or more selectors separated by ">".

The following rule sets the style of all P elements that are children of BODY:

```
body> P { line-height: 1.3 }
```

The following example combines descendant selectors and child selectors: *div ol>li p*

It matches a P element that is a descendant of an LI; the LI element must be the child of an OL element; the OL element must be a descendant of a DIV. Notice that the optional white space around the ">" combinator has been left out.

Adjacent sibling selectors:

Adjacent sibling selectors have the following syntax: *E1 + E2*, where E2 is the subject of the selector. The selector matches if E1 and E2 share the same parent in the document tree and E1 immediately precedes E2, ignoring non-element nodes (such as text nodes and comments).

Thus, the following rule states that when a P element immediately follows a MATH element, it should not be indented: *math + p { text-indent: 0 }*

The next example reduces the vertical space separating an H1 and an H2 that immediately follows it:

```
h1 + h2 { margin-top: -5mm }
```

The following rule is similar to the one in the previous example, except that it adds a class selector. Thus, special formatting only occurs when H1 has class="opener":

```
h1.opener + h2 { margin-top: -5mm }
```

Attribute selectors:

Matching attributes and attribute values

Attribute selectors may match in four ways:

```
[att]
```

Match when the element sets the "att" attribute, whatever the value of the attribute.

```
[att=val]
```

Match when the element's "att" attribute value is exactly "val".

```
[att~=val]
```

Represents an element with the att attribute whose value is a white space-separated list of words, one of which is exactly "val". If "val" contains white space, it will never represent anything (since the words are *separated* by spaces). If "val" is the empty string, it will never represent anything either.

```
[att|=val]
```

Represents an element with the att attribute, its value either being exactly "val" or beginning with "val" immediately followed by "-". This is primarily intended to allow language subcode matches (e.g., the hreflang attribute on the a element in HTML) as described in BCP 47 or its successor.

Attribute values must be identifiers or strings. The case-sensitivity of attribute names and values in selectors depends on the document language.

For example, the following attribute selector matches all H1 elements that specify the "title" attribute, whatever its value:

```
h1[title] { color: blue; }
```

In the following example, the selector matches all SPAN elements whose "class" attribute has exactly the value "example": *span[class=example] { color: blue; }*

Multiple attribute selectors can be used to refer to several attributes of an element, or even several times to the same attribute.

Here, the selector matches all SPAN elements whose "hello" attribute has exactly the value "Cleveland" and whose "goodbye" attribute has exactly the value "Columbus":

span[hello="Cleveland"][goodbye="Columbus"] { color: blue; }

The following selectors illustrate the differences between "=" and "~=". The first selector will match, for example, the value "copyright copyleft copyeditor" for the "rel" attribute. The second selector will only match when the "href" attribute has the value "http://www.nareshit.com/".

a[rel~="copyright"] a[href="http://www.enosislearning.com/"]

The following rule hides all elements for which the value of the "lang" attribute is "fr" (i.e., the language is French).

**[lang=fr] { display : none }*

The following rule will match for values of the "lang" attribute that begin with "en", including "en", "en-US", and "en-cockney":

**[lang|= "en"] { color : red }*

Similarly, the following aural style sheet rules allow a script to be read aloud in different voices for each role:

DIALOGUE[character=romeo] { voice-family: "Laurence Olivier", charles, male }

DIALOGUE[character=juliet] { voice-family: "Vivien Leigh", victoria, female }

Class selectors:

Working with HTML, authors may use the period (.) notation as an alternative to the ~= notation when representing the class attribute. Thus, for HTML, *div.value* and *div[class~=value]* have the same meaning. The attribute value must immediately follow the "period" (.).

For example, we can assign style information to all elements with *class~="pastoral"* as follows:

**.pastoral { color: green } /* all elements with class~=pastoral */* or just

.pastoral { color: green } / all elements with class~=pastoral */*

The following assigns style only to H1 elements with class~="pastoral":

H1.pastoral { color: green } / H1 elements with class~=pastoral */*

Given these rules, the first H1 instance below would not have green text, while the second would:

<H1>Not green</H1>

<H1 class="pastoral">Very green</H1>

To match a subset of "class" values, each value must be preceded by a ".".

For example, the following rule matches any P element whose "class" attribute has been assigned a list of space-separated values that includes "pastoral" and "marine":

p.marine.pastoral { color: green }

This rule matches when class="pastoral blue aqua marine" but does not match for class="pastoral blue".

ID selectors:

Document languages may contain attributes that are declared to be of type ID. What makes attributes of type ID special is that no two such attributes can have the same value; whatever the document language, an ID attribute can be used to uniquely identify its element. In HTML all ID attributes are named "id"; XML applications may name ID attributes differently, but the same restriction applies.

The ID attribute of a document language allows authors to assign an identifier to one element instance in the document tree. CSS ID selectors match an element instance based on its identifier. A CSS ID selector contains a "#" immediately followed by the ID value, which must be an identifier.

The following ID selector matches the H1 element whose ID attribute has the value "chapter1": *h1#chapter1 { text-align: center }*

In the following example, the style rule matches the element that has the ID value "z98y". The rule will thus match for the P element:

```
<head>
<title>match p</title>
<style type="text/css">
```

```

*#z98y { letter-spacing: 0.3em }
</style>
</head>
<body>
<p id=z98y>wide text</p>
</body>

```

In the next example, however, the style rule will only match an H1 element that has an ID value of "z98y". The rule will not match the P element in this example:

```

<head>
<title>match h1 only</title><style type="text/css">
h1#z98y { letter-spacing: 0.5em }
</style>
</head>
<body>
<p id=z98y>wide text</p>
</body>

```

ID selectors have a higher specificity than attribute selectors. For example, in HTML, the selector #p123 is more specific than [id=p123] in terms of the cascade.

If an element has multiple ID attributes, all of them must be treated as IDs for that element for the purposes of the ID selector.

CSS-Background

You can set the following background properties of an element:

The **background-color** property is used to set the background color of an element.

The **background-image** property is used to set the background image of an element.

The **background-repeat** property is used to control the repetition of an image in the background.

The **background-position** property is used to control the position of an image in the background.

The **background-attachment** property is used to control the scrolling of an image in the background.

The **background** property is used as a shorthand to specify a number of other background properties.

Animation Properties:

Property	Description	CSS Version
----------	-------------	-------------

@keyframes	Specifies the animation	3
animation	A shorthand property for all the animation properties below, except the animation-play-state property	3
animation-name	Specifies a name for the @keyframes animation	3
animation-duration	Specifies how many seconds or milliseconds an animation takes to complete one cycle	3
animation-timing-function	Specifies the speed curve of the animation	3
animation-delay	Specifies when the animation will start	3
animation-iteration-count	Specifies the number of times an animation should be played	3
animation-direction	Specifies whether or not the animation should play in reverse on alternate cycles	3
animation-play-state	Specifies whether the animation is running or paused	3

Background Properties:

Property	Description	CSS
background	Sets all the background properties in one declaration	1
background-	Sets whether a background image is fixed or scrolls with the rest of the	1
background-color	Sets the background color of an element	1
background-image	Sets the background image for an element	1
background-position	Sets the starting position of a background image	1
background-repeat	Sets how a background image will be repeated	1
background-clip	Specifies the painting area of the background	3
background-origin	Specifies the positioning area of the background images	3
background-size	Specifies the size of the background images	3

Border and Outline Properties:

Property	Description	CSS Version
----------	-------------	-------------

border	Sets all the border properties in one declaration	1
border-bottom	Sets all the bottom border properties in one declaration	1
border-bottom-color	Sets the color of the bottom border	1
border-bottom-style	Sets the style of the bottom border	1
border-bottom-width	Sets the width of the bottom border	1
border-color	Sets the color of the four borders	1
border-left	Sets all the left border properties in one declaration	1
border-left-color	Sets the color of the left border	1
border-left-style	Sets the style of the left border	1
border-left-width	Sets the width of the left border	1
border-right	Sets all the right border properties in one declaration	1
border-right-color	Sets the color of the right border	1
border-right-style	Sets the style of the right border	1
border-right-width	Sets the width of the right border	1
border-style	Sets the style of the four borders	1
border-top	Sets all the top border properties in one declaration	1
border-top-color	Sets the color of the top border	1
border-top-style	Sets the style of the top border	1
border-top-width	Sets the width of the top border	1
border-width	Sets the width of the four borders	1
outline	Sets all the outline properties in one declaration	2
outline-color	Sets the color of an outline	2
outline-style	Sets the style of an outline	2
outline-width	Sets the width of an outline	2
border-bottom-leftradius	Defines the shape of the border of the bottom-left corner	3
border-bottom-rightradius	Defines the shape of the border of the bottom-right corner	3
border-image	A shorthand property for setting all the border-image-	3
border-image-outset	Specifies the amount by which the border image area extends	3
border-image-repeat	Specifies whether the image-border should be repeated,	3
border-image-slice	Specifies the inward offsets of the image-border	3
border-image-source	Specifies an image to be used as a border	3
border-image-width	Specifies the widths of the image-border	3
border-radius	A shorthand property for setting all the four border-*radius	3
border-top-left-radius	Defines the shape of the border of the top-left corner	3
border-top-right-radius	Defines the shape of the border of the top-right corner	3
box-decoration-break		3
box-shadow	Attaches one or more drop-shadows to the box	3

Box Properties :

Property	Description	CSS Version
overflow-x	Specifies whether or not to clip the left/right edges of the content, if it overflows the element's content area	3
overflow-y	Specifies whether or not to clip the top/bottom edges of the content, if it overflows the element's content area	3
overflow-style	Specifies the preferred scrolling method for elements that overflow	3
rotation	Rotates an element around a given point defined by the rotation-point property	3
rotation-point	Defines a point as an offset from the top left border edge	3

Color Properties:

Property	Description	CSS Version
color-profile	Permits the specification of a source color profile other than the	3
opacity	Sets the opacity level for an element	3
rendering-intent	Permits the specification of a color profile rendering intent	3

Content for Paged Media Properties

Property	Description	CSS Version
bookmark-label	Specifies the label of the bookmark	3
bookmark-level	Specifies the level of the bookmark	3
bookmark-target	Specifies the target of the bookmark link	3
float-offset	Pushes floated elements in the opposite direction of the where	3
hyphenate-after	Specifies the minimum number of characters in a hyphenated	3
hyphenate-before	Specifies the minimum number of characters in a hyphenated	3
hyphenate-character	Specifies a string that is shown when a hyphenatebreak occurs	3
hyphenate-lines	Indicates the maximum number of successive hyphenated lines	3
hyphenate-resource	Specifies a comma-separated list of external resources that can	3
hyphens	Sets how to split words to improve the layout of paragraphs	3
image-resolution	Specifies the correct resolution of images	3
marks	Adds crop and/or cross marks to the document	3
string-set		3

Dimension Properties

Property	Description	CSS Version
----------	-------------	-------------

height	Sets the height of an element	1
max-height	Sets the maximum height of an element	2
max-width	Sets the maximum width of an element	2
min-height	Sets the minimum height of an element	2
min-width	Sets the minimum width of an element	2
width	Sets the width of an element	1

Flexible Box Properties:

Property	Description	CSS Version
box-align	Specifies how to align the child elements of a box	3
box-direction	Specifies in which direction the children of a box are displayed	3
box-flex	Specifies whether the children of a box is flexible or inflexible in	3
box-flex-group	Assigns flexible elements to flex groups	3
box-lines	Specifies whether columns will go onto a new line whenever it	3
box-ordinal-group	Specifies the display order of the child elements of a box	3
box-orient	Specifies whether the children of a box should be laid out	3
box-pack	Specifies the horizontal position in horizontal boxes and the	3

Font Properties:

Property	Description
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text
font-style	Specifies the font style for text
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-weight	Specifies the weight of a font
@font-face	A rule that allows websites to download and use fonts other than the "web-safe"
font-size-adjust	Preserves the readability of text when font fallback occurs
font-stretch	Selects a normal, condensed, or expanded face from a font family

Generated Content Properties:

Property	Description	CSS Version
----------	-------------	-------------

content	Used with the :before and :after pseudo-elements, to insert generated content	2
counter-increment	Increments one or more counters	2
counter-reset	Creates or resets one or more counters	2
crop	Allows a replaced element to be just a rectangular area of an object, instead of the whole object	3
move-to	Causes an element to be removed from the flow and reinserted at a later point in the document	3
page-policy	Determines which page-based occurrence of a given element is applied to a counter or string value	3

Grid Properties:

Property	Description	CSS Version
grid-columns	Specifies the width of each column in a grid	3
grid-rows	Specifies the height of each column in a grid	3

Hyperlink Properties:

Property	Description	CSS Version
target	A shorthand property for setting the target-name, target-new, and target-position properties	3
target-name	Specifies where to open links (target destination)	3
target-new	Specifies whether new destination links should open in a new window or in a new tab of an existing window	3
target-position	Specifies where new destination links should be placed	3

Linebox Properties:

Property	Description	CSS Version
alignment-adjust	Allows more precise alignment of elements	3
alignment-baseline	Specifies how an inline-level element is aligned with respect to its parent	3
baseline-shift	Allows repositioning of the dominant-baseline relative to the dominant-baseline	3
dominant-baseline	Specifies a scaled-baseline-table	3
drop-initial-after-adjust	Sets the alignment point of the drop initial for the primary connection point	3
drop-initial-after-align	Sets which alignment line within the initial line box is used at the primary connection point with the initial letter box	3
drop-initial-beforeadjust	Sets the alignment point of the drop initial for the secondary connection point	3
drop-initial-beforealign	Sets which alignment line within the initial line box is used at the secondary connection point with the initial letter box	3
drop-initial-size	Controls the partial sinking of the initial letter	3
drop-initial-value	Activates a drop-initial effect	3
inline-box-align	Sets which line of a multi-line inline block align with the previous and next inline elements within a line	3
line-stacking	A shorthand property for setting the line-stackingstrategy, line-stacking-ruby, and line-stacking-shift properties	3
line-stacking-ruby	Sets the line stacking method for block elements containing ruby annotation elements	3

line-stacking-shift	Sets the line stacking method for block elements containing elements with base-shift	3
line-stacking-strategy	Sets the line stacking strategy for stacked line boxes within a containing block element	3
text-height	Sets the block-progression dimension of the text content area of an inline box	3

List Properties:

Property	Description	CSS Version
list-style	Sets all the properties for a list in one declaration	1
list-style-image	Specifies an image as the list-item marker	1
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow	1
list-style-type	Specifies the type of list-item marker	1

Margin Properties:

Vis	Description	CSS Version
margin	Sets all the margin properties in one declaration	1
margin-bottom	Sets the bottom margin of an element	1
margin-left	Sets the left margin of an element	1
margin-right	Sets the right margin of an element	1
margin-top	Sets the top margin of an element	1

Marquee Properties

Property	Description	CSS Version
marquee-direction	Sets the direction of the moving content	3
marquee-play-count	Sets how many times the content move	3
marquee-speed	Sets how fast the content scrolls	3
marquee-style	Sets the style of the moving content	3

Multi-column Properties:

Property	Description	CSS Version
column-count	Specifies the number of columns an element should be divided into	3
column-fill	Specifies how to fill columns	3
column-gap	Specifies the gap between the columns	3
column-rule	A shorthand property for setting all the column-rule-* properties	3
column-rule-color	Specifies the color of the rule between columns	3
column-rule-style	Specifies the style of the rule between columns	3
column-rule-width	Specifies the width of the rule between columns	3
column-span	Specifies how many columns an element should span across	3
column-width	Specifies the width of the columns	3
columns	A shorthand property for setting column-width and column-count	3

Padding Properties:

Property	Description	CSS Version
padding	Sets all the padding properties in one declaration	1
padding-bottom	Sets the bottom padding of an element	1
padding-left	Sets the left padding of an element	1
padding-right	Sets the right padding of an element	1
padding-top	Sets the top padding of an element	1

Paged Media Properties

Property	Description	CSS Version
fit	Gives a hint for how to scale a replaced element if neither	3
fit-position	Determines the alignment of the object inside the box	3
image-orientation	Specifies a rotation in the right or clockwise direction that a	3
page	Specifies a particular type of page where an element	3
size	Specifies the size and orientation of the containing box for	3

Positioning Properties

Property	Description	CSS Version
bottom	Specifies the bottom position of a positioned element	2
clear	Specifies which sides of an element where other floating	1
clip	Clips an absolutely positioned element	2
cursor	Specifies the type of cursor to be displayed	2
display	Specifies how a certain HTML element should be displayed	1
float	Specifies whether or not a box should float	1
left	Specifies the left position of a positioned element	2
overflow	Specifies what happens if content overflows an element's box	2
position	Specifies the type of positioning method used for an element	2
right	Specifies the right position of a positioned element	2
top	Specifies the top position of a positioned element	2
visibility	Specifies whether or not an element is visible	2

z-index	Sets the stack order of a positioned element	2
----------------	--	---

Print Properties

Property	Description	CSS Version
orphans	Sets the minimum number of lines that must be left at the	2
page-break-after	Sets the page-breaking behavior after an element	2
page-break-before	Sets the page-breaking behavior before an element	2
page-break-inside	Sets the page-breaking behavior inside an element	2
widows	Sets the minimum number of lines that must be left at the	2

Ruby Properties:

Property	Description	CSS Version
ruby-align	Controls the text alignment of the ruby text and ruby base contents relative to each other	3
ruby-overhang	Determines whether, and on which side, ruby text is allowed to partially overhang any adjacent text in addition to its own base, when the ruby text is wider than the ruby base	3
ruby-position	Controls the position of the ruby text with respect to its base	3
ruby-span	Controls the spanning behavior of annotation elements	3

Speech Properties:

Property	Description	CSS Version
mark	A shorthand property for setting the mark-before and mark-	3
mark-after	Allows named markers to be attached to the audio	3
	stream	
mark-before	Allows named markers to be attached to the audio stream	3
phonemes	Specifies a phonetic pronunciation for the text contained by the	3
rest	A shorthand property for setting the rest-before and rest-after	3
rest-after	Specifies a rest or prosodic boundary to be observed after	3

rest-before	Specifies a rest or prosodic boundary to be observed before	3
voice-balance	Specifies the balance between left and right channels	3
voice-duration	Specifies how long it should take to render the selected	3
voice-pitch	Specifies the average pitch (a frequency) of the speaking voice	3
voice-pitch-range	Specifies variation in average pitch	3
voice-rate	Controls the speaking rate	3
voice-stress	Indicates the strength of emphasis to be applied	3
voice-volume	Refers to the amplitude of the waveform output by the speech	3

Table Properties

Property	Description	CSS Version
border-collapse	Specifies whether or not table borders should be collapsed	2
border-spacing	Specifies the distance between the borders of adjacent cells	2
caption-side	Specifies the placement of a table caption	2
empty-cells	Specifies whether or not to display borders and background on empty cells in a table	2
table-layout	Sets the layout algorithm to be used for a table	2

Text Properties:

Property	Description	CSS Version
color	Sets the color of text	1
direction	Specifies the text direction/writing direction	2
letter-spacing	Increases or decreases the space between characters in a text	1
line-height	Sets the line height	1
text-align	Specifies the horizontal alignment of text	1
text-decoration	Specifies the decoration added to text	1
text-indent	Specifies the indentation of the first line in a text-	1
	block	
text-transform	Controls the capitalization of text	1

unicode-bidi		2
vertical-align	Sets the vertical alignment of an element	1
white-space	Specifies how white-space inside an element is handled	1
word-spacing	Increases or decreases the space between words in a text	1
hanging-punctuation	Specifies whether a punctuation character may be placed outside the line box	3
punctuation-trim	Specifies whether a punctuation character should be trimmed	3
text-align-last	Describes how the last line of a block or a line right before a forced line break is aligned when text-align is "justify"	3
text-justify	Specifies the justification method used when text align is "justify"	3
text-outline	Specifies a text outline	3
text-overflow	Specifies what should happen when text overflows the containing element	3
text-shadow	Adds shadow to text	3
text-wrap	Specifies line breaking rules for text	3
word-break	Specifies line breaking rules for non-CJK scripts	3
word-wrap	Allows long, unbreakable words to be broken and wrap to the next line	3

2D/3D Transform Properties

Property	Description	Version
transform	Applies a 2D or 3D transformation to an element	3
transform-origin	Allows you to change the position on transformed elements	3
transform-style	Specifies how nested elements are rendered in 3D space	3
perspective	Specifies the perspective on how 3D elements are viewed	3
perspective-origin	Specifies the bottom position of 3D elements	3
backface-visibility	Defines whether or not an element should be visible when not facing the screen	3

Transition Properties:

Property	Description
transition	A shorthand property for setting the four transition properties
transition-property	Specifies the name of the CSS property the transition effect is for
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to
transition-timing-function	Specifies the speed curve of the transition effect
transition-delay	Specifies when the transition effect will start

User-interface Properties:

Property	Description
appearance	Allows you to make an element look like a standard user interface element
box-sizing	Allows you to define certain elements to fit an area in a certain way
icon	Provides the author the ability to style an element with an iconic equivalent
nav-down	Specifies where to navigate when using the arrowdown navigation key
nav-index	Specifies the tabbing order for an element
nav-left	Specifies where to navigate when using the arrow-left navigation key
nav-right	Specifies where to navigate when using the arrowright navigation key
nav-up	Specifies where to navigate when using the arrow-up navigation key
outline-offset	Offsets an outline, and draws it beyond the border edge
resize	Specifies whether or not an element is resizable by the user

Some Examples

1.The following example demonstrates how to use background image paroperties of css

```
<!doctype html>
```

```
<html>
```

```
<head><style> body
```

```
{
```

```
background-image:url(e:/images/garden.jpg);          background-repeat:no-repeat;
background-size:200px 200px; background-position:center center; background-
attachment:fixed;
```

```
}

</style>
</head>
<body>
<h1> Provide Text in the body to make the page scrollable </h1>
</body>
</html>
```

2.The following example animates a div element and moves it right and left with animation properties of css 3

```
<html>
<head>
<style> div
{
width:100px; height:100px; background:red; position:relative; animation:mymove 5s 2;
-webkit-animation:mymove 5s infinite; /*Safari and Chrome*/
}
@keyframesmymove
{
from {left:0px;} to {left:200px;}
}
@-webkit-keyframesmymove /*Safari and Chrome*/
{
from {left:0px;} to {left:200px;}
}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

3.The following example animates a div element and moves it right then down then left and finally top with animation properties of css3

```
<!doctype html>

<html>
```

```
<head>
```

```
<style> div
```

```
{
```

```
width:100px;
```

```
height:100px; background:red; position:relative; animation-name:myfirst; animation-
duration:5s; animation-timing-function:linear; animation-delay:2s; animation-iteration-
count:infinite; animation-direction:alternate; animation-play-state:running;
```

```
/* Firefox: */
```

```
-moz-animation-name:myfirst;
-moz-animation-duration:5s;
-moz-animation-timing-function:linear;
-moz-animation-delay:2s;
-moz-animation-iteration-count:infinite;
-moz-animation-direction:alternate;
-moz-animation-play-state:running;
/* Safari and Chrome: */
-webkit-animation-name:myfirst;
-webkit-animation-duration:5s;
-webkit-animation-timing-function:linear;
-webkit-animation-delay:2s;
-webkit-animation-iteration-count:infinite;
-webkit-animation-direction:alternate;
-webkit-animation-play-state:running;
/* Opera: */
-o-animation-name:myfirst;
-o-animation-duration:5s;
-o-animation-timing-function:linear;
-o-animation-delay:2s;
-o-animation-iteration-count:infinite;
-o-animation-direction:alternate;
-o-animation-play-state:running;
}
```

```
@keyframesmyfirst
```

```
{
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-keyframesmyfirst /* 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;}
}
@-webkit-keyframesmyfirst /* Safari and Chrome */
{
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;}
}
@-o-keyframesmyfirst /* Opera */
{
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>

```

4.The following example rotates an image when mouse is over the image with animation properties of css 3

```

<html>
<head>
<style>
@-webkit-keyframes spin
{
from { -webkit-transform: rotate(0deg);} to { -webkit-transform: rotate(360deg);}
}
@-moz-keyframes spin { from { -moz-transform: rotate(0deg);} to { -moz-transform:
rotate(360deg);}
}

```

```

@-o-keyframes spin { from { -o-transform: rotate(0deg);} to { -o-transform:
rotate(360deg);}
}
@-ms-keyframes spin { from { -ms-transform: rotate(0deg);} to { -ms-transform:
rotate(360deg);}
}
img
{
width:250px; height:250px; align:center;
}
img:hover
{
-webkit-animation: spin 5s infinite linear;
-moz-animation: spin 5s infinite linear;
-o-animation: spin 5s infinite linear;
-ms-animation: spin 5s infinite linear;
}
</style>
</head>
<body>

</body>
</html>

```

Css3 notes

CSS3 is a cascading piece of paper that specifies concerning the data with a joined hypertext markup language document display. it's considerably additional options than previous CSS versions. additionally to further graphics functions, CSS3 permits, to pick out additional hypertext markup language tags and outline however they're displayed on an online browser. The standard structure of CSS3 permits a gradual unharness of recent options, and lets browsers update piecemeal to support the most recent definitions.

CSS3 is completely backwards compatible, so you will not have to change existing designs. Browsers will always support **CSS2**. **CSS3** is split up into "modules". The old specification has been split into smaller pieces, and new ones have been also added.

CSS3 Animation:

An **Animation** 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 object into one style to another style in an animated way.

The all major browsers support **Animation** feature except Internet Explorer.

It gradually changes an object style to another style, The complete **Animation** depends upon the declaring the **Keyframes** with the **css3**.

CSS3 Borders:

A **CSS3 Border** is such an affords of style sheet which reduces the human efforts of Photoshop and other graphical applications. An individual can create the **rounded borders**, **border shadow**, **imaged based border** and etc. with the help of **CSS3 Border**.

Basically We use three features to create the border:

- ✓ border-radius
- ✓ box-shadow
- ✓ border-image

border-radius is a such property of CSS3 by which we can create the rounded corners.

box-shadow is a such property of CSS3 by which we can create the shadow of the border.

border-image is a such property of CSS3 by which we can create the customized border, as we can put our own image as a border.

CSS3 Fonts:

A **CSS3 Font** is an advance feature of **CSS3** which is used to improve the web designing. With the help of **CSS3 Fonts** feature we can create different types of font style.

The rule for defining the Fonts is only We have to declare a name in the first line of starting css properties. The font file can found in ttf(**True type font**) format or otf(**Open type font**) format.

CSS3 Multiple Columns:

A **Multiple Columns** is such an advance feature of **CSS3** which is used for creating the newspaper layout. You can create your articles to **Multiple Columns**, even if it is in one paragraph.

There are three properties of **CSS3 Multiple Columns** that is used to make the proper layout, what you want to do. The all major browsers support **Multiple Column** properties except Internet Explorer.

The three properties of **CSS3 Multiple Columns** have been described as follows:

- ✓ column-count
- ✓ column-rule
- ✓ column-gap

Where column-count defines the number of columns, column-rule defines the line style between the columns and the column-gap defines the gap (blank spaces) between the columns.

CSS3 Text Effects:

A **CSS3 Text Effect** is a such term which is used to implement some extra features on normal text. **CSS3 Text Effect** is used to extend the text features for viewing and layout purpose.

There are mainly two properties of **CSS3 Text Effects**, which have been described as follows:

- ✓ text-shadow
- ✓ word-wrap

Where text-shadow is used to create the shadow around the text, We can change the shadow color also. And word-wrap is used to break the continued text in another line. It means whenever we get difficulty to break the line of sentence we can generally use this `css3 text-wrap` property.

CSS3 Transition Effects:

A **CSS3 Transition Effect** is a such an effect that lets an element gradually change from one style to another style. **CSS3 Transition Effect** is best suited for animation uses. But still a lot can be done without using the animation. A user interface is necessary to see the effects of transition. The all major browser support the **CSS3 Transition Effects**.

Although CSS3 Transition Effect is sufficient for the transition of an element, but a text-transform property can enhance the style of CSS3 Transition Effects.

There are mainly four properties of **CSS3 Transition Effects**, which has been described as follows:

- ✓ transition-property
- ✓ transition-duration
- ✓ transition-timing-function
- ✓ transition-delay

transition-property

Where transition-property is used to define about the `css3` properties, on which the properties should be applied or not. The following Syntax can be found can be used to define the property.

transition-property: all;

transition-property: none;

transition-property: background-color;

transition-property: background-color, height, width;

transition-duration

Where transition-duration is used to define the time of corresponding transitions to take effect. The time can be set in seconds/milliseconds.

```
transition-duration: 2s;
transition-duration: 1000ms;
transition-duration: 1000ms, 2000ms;
```

transition-timing-function

Where transition-timing-function is used to define the style of transition take effect over its transition-duration. This can be done using the predefined function or can be done using a customized cubic process.

```
transition-timing-function: ease;
transition-timing-function: ease-in;
transition-timing-function: ease-in-out;
transition-timing-function: ease, linear;
transition-timing-function: cubic-bezier(1.000, 0.835, 0.000, 0.945);
```

transition-delay

Where transition-delay is used to determine the time duration between transition start and it finishing. Negative values are also acceptable in transition-delay.

```
transition-delay: 2s;
transition-delay: 1000ms, 2000ms;
transition-delay: -2s;
```

CSS3 User Interface:

CSS3 User Interface is not much popular features of CSS3. It is also not a milestone in designing that it shouldn't be avoided. But after all that if you want, you can manipulate these user interfaces to enhance the design skills.

Css3 has introduced mainly three types of user interface that has been described as follows:

- ✓ resize
- ✓ box-sizing
- ✓ outline-offset

The `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`.

CSS3 2D Transform:

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.

CSS3 2D Transform has introduced mainly five types of methods that has been described as follows:

- ✓`translate()`
- ✓`rotate()`
- ✓`scale()`
- ✓`skew()`
- ✓`matrix()`

`translate()` method

The **`translate()` Method** is used to move an object depending on its parameter. Two types of parameter you can pass in this method one is from left (x-axis) and the other is from the top (y-axis).

CSS3 3D Transform:

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()`

Bootstrap-notes

What is Bootstrap?

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs

What is Responsive Web Design?

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

Bootstrap History

Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter, and released as an open source product in August 2011 on GitHub.

In June 2014 Bootstrap was the No.1 project on GitHub!

Why Use Bootstrap?

Advantages of Bootstrap:

- **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
- **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
- **Mobile-first approach:** In Bootstrap 3, mobile-first styles are part of the core framework
- **Browser compatibility:** Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Safari, and Opera)

Where to Get Bootstrap?

There are two ways to start using Bootstrap on your own web site.

You can:

- Download Bootstrap from getbootstrap.com
- Include Bootstrap from a CDN

Downloading Bootstrap

If you want to download and host Bootstrap yourself, go to getbootstrap.com, and follow the instructions there.

Bootstrap CDN:

If you don't want to download and host Bootstrap yourself, you can include it from a CDN (Content Delivery Network).

What is Bootstrap Grid System?

Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases. It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.

Let us understand the above statement. Bootstrap 3 is mobile first in the sense that the code for Bootstrap now starts by targeting smaller screens like mobile devices, tablets, and then “expands” components and grids for larger screens such as laptops, desktops.

Mobile First Strategy

- **Content**
 - Determine what is most important.
- **Layout**
 - Design to smaller widths first.
 - Base CSS address mobile device first; media queries address for tablet, desktops.
- **Progressive Enhancement**
 - Add elements as screen size increases.

Working of Bootstrap Grid System:

Grid systems are used for creating page layouts through a series of rows and columns that house your content. Here's how the Bootstrap grid system works –

- Rows must be placed within a **.container** class for proper alignment and padding.
- Use rows to create horizontal groups of columns.

- Content should be placed within the columns, and only columns may be the immediate children of rows.
- Predefined grid classes like **.row** and **.col-xs-4** are available for quickly making grid layouts. LESS mixins can also be used for more semantic layouts.
- Columns create gutters (gaps between column content) via padding. That padding is offset in rows for the first and the last column via negative margin on **.rows**.
- Grid columns are created by specifying the number of twelve available columns you wish to span. For example, three equal columns would use three **.col-xs-4**.

Media Queries

Media query is a really fancy term for "conditional CSS rule". It simply applies some CSS, based on certain conditions set forth. If those conditions are met, the style is applied.

Media Queries in Bootstrap allow you to move, show and hide content based on the viewport size. Following media queries are used in LESS files to create the key breakpoints in the Bootstrap grid system.

```
/* Extra small devices (phones, less than 768px) */
/* No media query since this is the default in Bootstrap */

/* Small devices (tablets, 768px and up) */
@media (min-width: @screen-sm-min) { ... }

/* Medium devices (desktops, 992px and up) */
@media (min-width: @screen-md-min) { ... }

/* Large devices (large desktops, 1200px and up) */
@media (min-width: @screen-lg-min) { ... }
```

Occasionally these are expanded to include a **max-width** to limit CSS to a narrower set of devices.

```
@media (max-width: @screen-xs-max) { ... }
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }
@media (min-width: @screen-md-min) and (max-width: @screen-md-max) { ... }
@media (min-width: @screen-lg-min) { ... }
```

Media queries have two parts, a device specification and then a size rule. In the above case, the following rule is set –

Let us consider this line –

```
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }
```

For all devices no matter what kind with *min-width: @screen-sm-min* if the width of the screen gets smaller than *@screen-sm-max*, then do something.

Grid options :The following table summarizes aspects of how Bootstrap grid system works across multiple devices –

	Extra small devices (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints
Max container width	None (auto)	750px	970px	1170px
Class prefix	.col-xs-	.col-sm-	.col-md-	.col-lg-
# of columns	12	12	12	12
Max column width	Auto	60px	78px	95px
Gutter width	30px (15px on each side of a column)	30px (15px on each side of a column)	30px (15px on each side of a column)	30px (15px on each side of a column)
Nestable	Yes	Yes	Yes	Yes
Offsets	Yes	Yes	Yes	Yes

Column ordering	Yes	Yes	Yes	Yes
-----------------	-----	-----	-----	-----

Basic Grid Structure

Following is basic structure of Bootstrap grid –

```
<div class="container">

<div class="row">
<div class="col-*-*"></div>
<div class="col-*-*"></div>
</div>

<div class="row">...</div>

</div>

<div class="container">
....
</div>
```

Bootstrap allows you to display code with two different key ways –

- The first is the `<code>` tag. If you are going to be displaying code inline, you should use the `<code>` tag.
- Second is the `<pre>` tag. If the code needs to be displayed as a standalone block element or if it has multiple lines, then you should use the `<pre>` tag.

Make sure that when you use the `<pre>` and `<code>` tags, you use the unicode variants for the opening and closing tags – `<`; and `>`;

Let us see an example below –

```
<p><code>&lt;header&gt;</code> is wrapped as an inline element.</p>
<p>To display code as a standalone block element use &lt;pre&gt; tag as:</p>

<pre>
```

```
<article>
<h1>Article Heading</h1>
</article>
</pre>
```

Bootstrap provides a clean layout for building tables. Some of the table elements supported by Bootstrap are –

Tag	Description
<table>	Wrapping element for displaying data in a tabular format
<thead>	Container element for table header rows (<tr>) to label table columns.
<tbody>	Container element for table rows (<tr>) in the body of the table.
<tr>	Container element for a set of table cells (<td> or <th>) that appears on a single row.
<td>	Default table cell.
<th>	Special table cell for column (or row, depending on scope and placement) labels. Must be used
<caption>	Description or summary of what the table holds.

Basic Table:

If you want a nice, basic table style with just some light padding and horizontal dividers, add the base class of *.table* to any table as shown in the following example –

```
<table class="table">
<caption>Basic Table Layout</caption>
```

```
<thead>
<tr>
<th>Name</th>
<th>City</th>
</tr>
```



```

</thead>
<tbody>
<tr>
<td>Tanmay</td>
<td>Bangalore</td>
</tr>
<tr>
<td>Sachin</td>
<td>Mumbai</td>
</tr>
</tbody>
</table>

```

Optional Table Classes

Along with the base table markup and the `.table` class, there are a few additional classes that you can use to style the markup. Following sections will give you a glimpse of all these classes.

Striped Table

By adding the `.table-striped` class, you will get stripes on rows within the `<tbody>` as seen in the following example –

```

<table class="table table-striped">
<caption>Striped Table Layout</caption>

<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Pincode</th>
</tr>
</thead>

<tbody>
<tr>
<td>Tanmay</td>
<td>Bangalore</td>
<td>560001</td>
</tr>

<tr>

```

```

<td>Sachin</td>
<td>Mumbai</td>
<td>400003</td>
</tr>

<tr>
<td>Uma</td>
<td>Pune</td>
<td>411027</td>
</tr>
</tbody>

</table>

```

Bordered Table

By adding the *.table-bordered* class, you will get borders surrounding every element and rounded corners around the entire table as seen in the following example –

```

<table class="table table-bordered">
<caption>Bordered Table Layout</caption>

<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Pincode</th>
</tr>
</thead>

<tbody>
<tr>
<td>Tanmay</td>
<td>Bangalore</td>
<td>560001</td>
</tr>

<tr>
<td>Sachin</td>
<td>Mumbai</td>
<td>400003</td>
</tr>

<tr>
<td>Uma</td>
<td>Pune</td>
<td>411027</td>
</tr>
</tbody>

```

```
</table>
```

Bootstrap Form:

Bootstrap provides you with following types of form layouts –

- Vertical (default) form
- In-line form
- Horizontal form

Vertical or Basic Form

The basic form structure comes with Bootstrap; individual form controls automatically receive some global styling. To create a basic form do the following –

- Add a role *form* to the parent `<form>` element.
- Wrap labels and controls in a `<div>` with class *.form-group*. This is needed for optimum spacing.
- Add a class of *.form-control* to all textual `<input>`, `<textarea>`, and `<select>` elements.

```
<form role="form">

<div class="form-group">
<label for="name">Name</label>
<input type="text" class="form-control" id="name" placeholder="Enter Name">
</div>

<div class="form-group">
<label for="inputfile">File input</label>
<input type="file" id="inputfile">
<p class="help-block">Example block-level help text here.</p>
</div>

<div class="checkbox">
<label><input type="checkbox"> Check me out</label>
</div>

<button type="submit" class="btn btn-default">Submit</button>
</form>
```

Inline Form

To create a form where all of the elements are inline, left aligned and labels are alongside, add the class *.form-inline* to the `<form>` tag.

```
<form class="form-inline" role="form">

<div class="form-group">
<label class="sr-only" for="name">Name</label>
<input type="text" class="form-control" id="name" placeholder="Enter Name">
</div>

<div class="form-group">
<label class="sr-only" for="inputfile">File input</label>
<input type="file" id="inputfile">
</div>

<div class="checkbox">
<label><input type="checkbox"> Check me out</label>
</div>

<button type="submit" class="btn btn-default">Submit</button>
</form>
```

- By default inputs, selects, and textareas have 100% width in Bootstrap. You need to set a width on the form controls when using inline form.
- Using the class *.sr-only* you can hide the labels of the inline forms.

Horizontal Form

Horizontal forms stands apart from the others not only in the amount of markup, but also in the presentation of the form. To create a form that uses the horizontal layout, do the following

- Add a class of *.form-horizontal* to the parent `<form>` element.
- Wrap labels and controls in a `<div>` with class *.form-group*.
- Add a class of *.control-label* to the labels.

```
<form class="form-horizontal" role="form">

<div class="form-group">
<label for="firstname" class="col-sm-2 control-label">First Name</label>

<div class="col-sm-10">
<input type="text" class="form-control" id="firstname" placeholder="Enter First Name">
</div>
</div>

<div class="form-group">
```

```

<label for="lastname" class="col-sm-2 control-label">Last Name</label>

<div class="col-sm-10">
<input type="text" class="form-control" id="lastname" placeholder="Enter Last Name">
</div>
</div>

<div class="form-group">
<div class="col-sm-offset-2 col-sm-10">
<div class="checkbox">
<label><input type="checkbox"> Remember me</label>
</div>
</div>
</div>

<div class="form-group">
<div class="col-sm-offset-2 col-sm-10">
<button type="submit" class="btn btn-default">Sign in</button>
</div>
</div>

</form>

```

Bootstrap-button:

Bootstrap provides some options to style buttons, which are summarized in the following table

–

Class	Description
Btn	Default/ Standard button.
btn-primary	Provides extra visual weight and identifies the primary action in a set of buttons.
btn-success	Indicates a successful or positive action.
btn-info	Contextual button for informational alert messages.
btn-warning	Indicates caution should be taken with this action.
btn-danger	Indicates a dangerous or potentially negative action.
btn-link	Deemphasize a button by making it look like a link while maintaining button behavior.

Button Size

The following table summarizes the classes used to get buttons of various sizes –

Class	Description
.btn-lg	This makes the button size large.
.btn-sm	This makes the button size small.
.btn-xs	This makes the button size extra small.
.btn-block	This creates block level buttons—those that span the full width of a parent.

Bootstrap-images:

Bootstrap provides three classes that can be used to apply some simple styles to images –

- *.img-rounded* – adds *border-radius:6px* to give the image rounded corners.
- *.img-circle* – makes the entire image round by adding *border-radius:500px*.
- *.img-thumbnail* – adds a bit of padding and a gray border –

The following example demonstrates this –

```



```

JavaScript

JavaScript is a scripting language designed and first implemented by Netscape (with help from Sun Microsystems). JavaScript was created by Brendan Eich at Netscape in 1995 for the purpose of allowing code in webpages. Netscape first introduced a JavaScript interpreter in Navigator 2. The interpreter was an extra software component in the browser that was capable of interpreting JavaScript source code inside an HTML document. This meant that Web page authors using no more than a simple text editor could place special instructions or programs directly inside Web pages to make them more dynamic and interactive. JavaScript can be used to:

- validate user input in an HTML form before sending the data to a server
- build forms that respond to user input without accessing a server
- change the appearance of HTML documents and dynamically write HTML into separate Windows
- open and close new windows or frames
- manipulate HTML "layers" including hiding, moving, and allowing the user to drag them around a browser window
- build small but complete client side programs

JavaScript Features:

- Java Script is a scripting language and it is not java.
- Java script is interpreter based scripting language.
- Java script is case sensitive.
- Java script is object based language as it provides predefined objects.
- Every statement in java script must be terminated with ;
- Most of the java script control statements syntax is same as syntax of control statements in C language.

Datatypes and Variable Declaration:

Java script didn't provide any data types for declaring variables and a variable in java script can store any type of value. Hence java script is loosely typed language. You can use a variable directly without declaring it as follows.

```
X = 10;
```

You can also declare a variable and then use it and for this use the keyword *var* as follows

```
Var x = 10;
```

Or

```
Var x;
```

```
X=10;
```

Writing JavaScript:

Java script can be written either as inline java script or internal java script or external java script.

Inline Java Script

When java script was written within the html element using attributes related to events of the element then it is called as inline java script.

JavaScriptDialogBoxes

Alert() : Alert function of java script is used to give an alert message to the user.

Prompt() : Prompt function of java script is used to prompt for a value from the user.

Confirm() : confirm function of java script is used to get confirmation from user before executing some task.

Example : The following example gives an alert message to the user when user click on the button using inline java script.

```
<!doctype html>

<html>

<form>

<input type="button" value="Click" onclick="alert('Button Clicked')"/>

</form>

</html>
```

Internal Java Script

When java script was written within the <head> section using <script> element then it is called as internal java script. The <script> element syntax is as follows.

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

</script>
```

Within the <head> section java script will be written in the form of functions. To create a function in java script use the keyword *function* and it has the following syntax.

Defining Functions Within Scripts

An important part of JavaScript is the ability to create new functions within scripts. These functions are named segments of JavaScript statements. These statements usually have a single purpose, such as performing a complex calculation or verifying the data entered into an HTML form. Functions can be passed copies of variables or references to objects to work with. Just as with built in functions, this is done in the parameter list. JavaScript functions that you define can be placed inside script tags anywhere in the document. However, they should be placed in the head of the document to guarantee they are loaded before being called from script statements in the body of the document. Here is the format for defining a JavaScript function:

```
functionfunctionName(parameter_1, parameter_2, .. parameter_n)
{
    statement1 statement2 statementn
}
```

The keyword `function` precedes the name of the function. A function name is like a variable name in that it must start with a character and must not be the same as a JavaScript key word. The parameter list is a comma separated list of variable names inside round brackets immediately following the function name. The statements to be executed when the function is called are contained inside the function body and are contained within curly braces.

Here is a function that finds the maximum value of two numbers. Note that the function determines which number is largest and returns the largest value using the `return` statement. The value returned will take the place of the function in the calling expression - see the example below.

```
functiongetMax(n1, n2)

{    if (n1 < n2)        return n2

    else        return n1

}
```

Example : The following example displays an alert message after prompting name of the user when user click on the button using internal java script.

```
<!doctype html>
<html>
<head>
<script>
```

```

Function getname()
{
Name=prompt('Enter Your Name');
Alert('Welcome Mr/Mrs ' + name);
}
</script>
</head>
<form>
<input type="button" value="Click" onclick="getname()"/>
</form>
</html>

```

ExternalJavaScript

Writing java script in a separate file with extension .js is called as external java script. For adding the reference of an external java script file to your html page, use <script> tag with src attribute as follows

```
<script type="text/javascript" src="filename.js"/>
```

Example : the following example demonstrates how to create and use external java script.

Create a file with name functions.js and write the following java script functions in it.

```

function sum(x,y)
{
returnparseInt(x) + parseInt(y);
}
function sub(x,y)
{
returnparseInt(x) - parseInt(y);
}
functionmul(x,y)
{
returnparseInt(x) * parseInt(y);
}
function div(x,y)
{
returnparseInt(x) / parseInt(y);
}

```

Create a html page and use the file functions.js as follows

```

<!doctype html>
<html>
<head>
<script src="d:\functions.js">

```

```

</script>
</head>
<body>
<form>
<table>
<tr>
<td> First Number </td>
<td><input type="text" name="t1"/></td>
</tr>
<tr>
<td> Second Number </td>
<td><input type="text" name="t2"/></td>
</tr>
<tr>
<td> Result </td>
<td><input type="text" name="t3"/></td>
</tr>
<tr>
<td colspan="2">
<input type="button" value="sum" onclick="t3.value=sum(t1.value,t2.value)"/>&nbsp;
<input type="button" value="sub" onclick="t3.value=sub(t1.value,t2.value)"/>&nbsp;
<input type="button" value="mul" onclick="t3.value=mul(t1.value,t2.value)"/>&nbsp;
<input type="button" value="div" onclick="t3.value=div(t1.value,t2.value)"/>&nbsp;
</td>
</tr>
</table>
</form>
</body>
</html>

```

Switch

Switch(<expression>)

```

{
Case <result1> :
    <stmts>
    Break;
Case <result2>:
    <stmts>
    Break;
.....
Default :
    <stmts>
}

```

TernaryOperator(?:)

```
<var> = (<condition>)?<stmt1>:<stmt2>;
```

LoopingControlStatements**While**

```
<var initialization>
While(<condition>)
{
    <stmts>
    <Var increment / decrement>
}
```

For

```
For(initialization;condition;incr/decr)
{
    <stmts>
}
```

Dowhile

```
<variable initialization>
Do
{
    <stmts>
    <var increment / decrement> }while(<condition>;
```

WorkingWithArrays:

An array is a very common data structure - a way of organizing and accessing data. It is supported by almost every computer programming language. Data such as numbers are stored and accessed as though they were in a list. Usually, arrays are created by some kind of declaration that indicates the type of data being stored in each "slot" or "box" in the array and how many "slots" the array should have. JavaScript 1.1, introduced with Navigator 3 included a new prebuilt array object. To create an array you use new Array(n) where n was the number of slots in the array or new Array() omitting the size. For example, using the new keyword Array:

```
mynumbers = new Array(5)    mynumbers[0] = 34
mynumbers[1] = 22           mynumbers[2] = 9
mynumbers[3] = 12 mynumbers[4] = 0
```

```
sum = 0;    for (i=0; i<mynumbers.length; i++)
{
    sum = sum + mynumbers[i];
```

```
}    alert(sum)
```

The new array object had the advantage that you could create the array and add values at the same time:

```
mynumbers = new Array(34, 22, 9, 12, 0)
```

This made it possible to do in one line what it took six lines to do in the previous example.

The one problem with this was that `mynumbers = new Array(2, 1)` creates an array with two values in it but `mynumbers = new Array(2)` creates an array with two empty slots. Logically, it should create one slot with the number 2 in it.

Example : The following example demonstrates how to work with one dimensional arrays

```
<html>
<head>
<script type="text/javascript">var A=new Array(5);
for(i=0;i<5;i++)
{
A[i]=eval(prompt("Enter An Integer"));
}
for(i=0;i<5;i++)
{
document.write("<h1>" + A[i] + "</h1>");
}
</script>
</head><body>
</body>
</html>
```

TwoDimensionalArrays

Two dimensional arrays in java script are declared using the following syntax.

```
var x = new Array(10); for (vari = 0; i < 10; i++)
```

```
{
x[i] = new Array(20);
}
x[5][12] = 3.0;
```

Example : The following example demonstrates how to use two dimensional arrays.

```
<html>
<head>
<script type="text/javascript">

var A = new Array(3);
for (vari = 0; i < 3; i++)
{
A[i] = new Array(3);
}
for(i=0;i<3;i++)
{
```

[illegible]

Example : The following example demonstrates how to use Array *Concat()* Method. This method concatenates the elements of one array at the end of another array and returns an array.

```
<html>
<head>
<script type="text/javascript">

var A=new Array(23,45,17,31,59); var B=new Array("a","b","c"); var
C=B.concat(A); document.write("<h1>" + C.toString() + "</h1>");

</script>
</head><body>

</body>
</html>
```

Example : The following example demonstrates how to use *Array Sort() and Reverse()* Methods. Sort() methods sorts the elements in array and reverse() methods reverses the elements.

```
<html>
<head>
<script type="text/javascript">

function compare(a,b)
{
return a-b;
}
```

```

var A=new Array(23,45,17,31,59,130); var B=A.sort(compare);
document.write("<h1>" + B.toString() + "</h1>");
document.write("<h1>" + B.reverse() + "</h1>");
</script>
</head><body>

</body>
</html>

```

Example : The following example demonstrates how to use Array *Slice()* Method. Slice method will extract specified number of elements starting from specified index without deleting them from array.

```

<html>
<head>
<script type="text/javascript">

var A=new Array(10,20,30,40,50,60,70); var B=A.slice(1,4);
document.write("<h1>" + A.toString() + "</h1>");
document.write("<h1>" + B.toString() + "</h1>");
</script>
</head>
<body>
</body>
</html>

```

Example : The following example demonstrates how to use Array *Splice()* Method. Splice method will extract specified number of elements starting from specified index and deletes them from array.

```

<html>
<head>
<script type="text/javascript">

var A=new Array(10,20,30,40,50,60,70); var B=A.splice(1,4);
document.write("<h1>" + A.toString() + "</h1>");
document.write("<h1>" + B.toString() + "</h1>");
</script>
</head><body>

</body>
</html>

```

Example : The following example demonstrates how to use Array *Splice()* Method. Splice method can extract specified number of elements starting from specified index and replace them with other elements.

```

<html>
<head>
<script type="text/javascript">

```

```
var A=new Array(10,20,30,40,50,60,70); var B=A.splice(1,4,1,2,3,4);
document.write("<h1>" + A.toString() + "</h1>");
document.write("<h1>" + B.toString() + "</h1>");
</script>
```

```
</head><body>
```

```
</body>
```

```
</html>
```

Example : The following example demonstrates how to use Array *Push()* and *Pop()* Methods. Push method pushes element at the top and pop elements pops the top element.

```
<html>
<head>
<script type="text/javascript">var A=new
Array(10,20,30,40,50);

A.push(eval(prompt("Enter Value to push"))); document.write("<h1>" +
A.toString() + "</h1>"); A.push(eval(prompt("Enter Value to push")));
document.write("<h1>" + A.toString() + "</h1>"); var p=A.pop();
document.write("<h1>Element Popped Is " + p + "</h1>"); var p=A.pop();
document.write("<h1>Element Popped Is " + p + "</h1>");
document.write("<h1>" + A.toString() + "</h1>");
</script>
</head><body>
```

```
</body>
```

```
</html>
```

Example : The following example demonstrates how to use Array *Sort()* method for sorting strings.

```
<html>
<head>
<script type="text/javascript"> function compare(a,b)

{return a.localeCompare(b);
}
var A=new Array('Microsoft','Adobe','Oracle','Sun','IBM'); document.write("<h1>" + A.sort(compare) +
"</h1>");

</script>
</head><body>

</body>
</html>
```


Validations

One main purpose of java script is performing validations. The following examples demonstrates how to perform various validations in java script.

Example : The following example demonstrates how to restrict a text box to accept only digits.

```
<html>
<head>
<script      type="text/javascript">      function

fn_validateNumeric(thi,dec)

{
/* verifying the character pressed is a digit */ if (((event.keyCode< 48) ||

(event.keyCode> 57)) &&

(event.keyCode != 46))
{
event.returnValue = false;
}
/* determining whether or not allow decimal point(.) if second argument is "n" then decimal point not
allowed. else part will check whether the decimal point (.) is pressed second time and if it is second time
then not allowed*/ if(dec=="n" &&event.keyCode == 46)
{
event.returnValue = false;
}
else
{
if(event.keyCode == 46 &&thi.value.indexOf(".")>=0)
{
event.returnValue = false;
}
}
}
/* Ascii values of 0-9 are 48 to 57 and ascii value of "." is 46. "This" refers to the control that causes the
event. else condition is to restrict "." more than once */
</script>
</head>
<body>
```

```

<input type="text" onkeypress=
"fn_validateNumeric(this,'y')" name="t1"/>
</body>
</html>

```

Example : The following example demonstrates how to restrict a text box to accept only alphabets.

```

<html>
<head>
<script type="text/javascript"> function
fn_validateAlpha(thi)
{
if (((event.keyCode< 65) || (event.keyCode> 90)) && ((event.keyCode<97) || (event.keyCode>122)))
{
event.returnValue = false;
}
}
</script>
</head>
<body>
<input type="text" onkeypress="fn_validateAlpha(this)" name="t1"/>
</body>
</html>

```

Example : The following example demonstrates how to automatically convert the letters typed in a text box to uppercase.

```

<html>
<head>
<script type="text/javascript">
functionConvert_Capital(thi)
{
if ((event.keyCode>=97) && (event.keyCode<=122))
[
event.keyCode=event.keyCode-32;
}
}
</script>
</head>
<body>
<input type="text" onkeypress="Convert_Capital(this)" name="t1"/>
</body>
</html>

```

Example : The following example demonstrates how to perform various validations like textbox is not blank , password is minimum 8 characters, password and confirm password are same and email id is in correct format.

```

<html>
<head>
<script type="text/javascript"> function emailcheck()

{
varstr=f1.txtemail.value;  var  at="@";  var

dot=".";                  varatpos=str.indexOf(at);

varlen=str.length; vardotpos=str.indexOf(dot); if

(atpos== -1 || atpos==0 || atpos==len)

{
alert("Invalid E-mail ID"); return false;

}

if (dotpos== -1 || dotpos==0 || dotpos==len)
{
alert("Invalid E-mail ID"); return false;

}

if (str.indexOf(at,(atpos+1))!= -1)
{
alert("Invalid E-mail ID"); return false;

}

/* checking immediate before character of @ or immediate next character of @ is .
*/ if (str.substring(atpos-1,atpos)==dot || str.substring(atpos+1,atpos+2)==dot)

{
alert("Invalid E-mail ID"); return false;

}

/* Checking .exists next to @ symbol or not */ if

(str.indexOf(dot,(atpos+2))!= -1)

{
alert("Invalid E-mail ID"); return false;

}

if (str.indexOf(" ")!= -1)
{
alert("Invalid E-mail ID"); return false;

```

```

}
return true;
}
functionisvalid()
{
if(f1.txtuname.value=="")
{
alert("You      Must      Enter      User      Name");

f1.txtuname.focus(); return false;

}
if(f1.ttxpwd1.value==" " || f1.ttxpwd2.value==" ")
{
alert("You Must Enter Password And Re Enter Password"); f1.ttxpwd1.focus();

return false;

}
if(f1.ttxpwd1.value.length<8)
{
alert("Password      Must      Be      Minimum      8      characters");

f1.ttxpwd1.value=""; f1.ttxpwd2.value=""; f1.ttxpwd1.focus(); return

false;

}
if(f1.ttxpwd1.value!=f1.ttxpwd2.value)
{
alert("Passsword and Re-Enter Password Must Be Same"); f1.ttxpwd1.value="";

f1.ttxpwd2.value=""; f1.ttxpwd1.focus(); return false;

}
if(f1.txtemail.value=="")
{
alert("You Must Enter Email Id"); f1.txtemail.focus();

return false;

}
if(emailcheck()==false)
{

```

```

f1.txtemail.value="";    f1.txtemail.focus();

return false;

}
return true;
}
</script>
</head>
<body>
<form id="f1">
<table>
<tr>
<td> Enter User Name </td>
<td><input type="text" name="txtuname" /></td>
</tr>
<tr>
<td> Enter Password </td>
<td><input type="password" name="txtpwd1"/></td>
</tr>
<tr>
<td> Re-Enter Password </td>
<td><input type="password" name="txtpwd2"/></td>
</tr>
<tr>
<td> Enter Email id </td>
<td><input type="text" name="txtemail"/></td>
</tr>
<tr>
<td colspan="2" align="right"><input type="button"        name="btnregister"
value="Register" onclick="isvalid()"/></td>
</tr>
</table>
</form>
</body>
</html>

```

Cookies

Web Browser and Server use HTTP protocol to communicate and HTTP is a stateless protocol. But for a commercial website it is required to maintain session information among different pages. For example one user registration ends after completing many pages. But how to maintain user's session information across all the web pages. In many situations, using cookies is the most efficient method of remembering and tracking

preferences, purchases, commissions, and other information required for better visitor experience or site statistics.

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.

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.

Cookies were originally designed for CGI programming and cookies' data is automatically transmitted between the web browser and web server, so CGI scripts on the server can read and write cookie values that are stored on the client. JavaScript can also manipulate cookies using the cookie property of the Document object. JavaScript can read, create, modify, and delete the cookie or cookies that apply to the current web page.

Storing Cookies

The simplest way to create a cookie is to assign a string value to the document.cookie object, which looks like this: Syntax:

```
document.cookie = "key1=value1;key2=value2;expires=date";
```

Here expires attribute is option. If you provide this attribute with a valid date or time then cookie will expire at the given date or time and after that cookies' value will not be accessible.

Note: Cookie values may not include semicolons, commas, or whitespace. For this reason, you may want to use the JavaScript escape() function to encode the value before storing it in the cookie. If you do this, you will also have to use the corresponding unescape() function when you read the cookie value.

Example : The following example demonstrates how to work with cookies for storing user id and password in cookies in a login form.

```

<!doctype html>
<html>
<head><script> function loginclick()

{
var c=document.getElementsByName("chkremember")[0];
if(c.checked)
{
setCookie("uid1",f1.txtuid.value,7);
}
}
functiongetCookie(c_name)
{
var i,x,y,ARRcookies=document.cookie.split(";");

for
(i=0;i<ARRcookies.length;i++)

{
x=ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));

y=ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1); if (x==c_name)

{
returnunescape(y);
}
}
}
functionsetCookie(c_name,value,exdays)
{
varexdate=new Date(); exdate.setDate(exdate.getDate() + exdays);

varc_value= escape(value) + ((exdays==null) ? "" : ";
expires="+exdate.toUTCString()); document.cookie=c_name + "=" + c_value;

}
functioncheckCookie()
{
var username=getCookie("uid1"); alert(username); if

(username!=null && username!="")

{

```

```

return username;
}
}
</script>
</head>
<body onload="f1.txtuid.value=checkCookie()">
<form id="f1">
<table>
<tr>
<td> User Name </td>
<td><input type="text" name="txtuid"/></td>
</tr>
<tr>
<td> Password </td>
<td><input type="password" name="txtpwd"/></td>
</tr>
<tr>
<td colspan="2"><input type="checkbox" name="chkremember"/> Remember Me
</td>
</tr>
<tr>
<td colspan="2" align="right"> <input type="button" name="btnlogin" value="login"
onclick="loginclick()"/></td>
</tr>
</table>
</form>
</body>
</html>

```

DOM(DocumentObjectModel)“Window”Object

The window object represents an open window in a browser. If a document contain frames (<frame> or <iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.

DOM(DocumentObjectModel)“Window”ObjectProperties

Property	Description
closed	Returns a Boolean value indicating whether a window has been closed or not

defaultStatus	Sets or returns the default text in the statusbar of a window
document	Returns the Document object for the window (See Document object)
frames	Returns an array of all the frames (including iframes) in the current window
history	Returns the History object for the window (See History object)
innerHeight	Sets or returns the inner height of a window's content area
innerWidth	Sets or returns the inner width of a window's content area
length	Returns the number of frames (including iframes) in a window
location	Returns the Location object for the window (See Location object)
name	Sets or returns the name of a window
navigator	Returns the Navigator object for the window (See Navigator object)
opener	Returns a reference to the window that created the window
outerHeight	Sets or returns the outer height of a window, including toolbars/scrollbars
outerWidth	Sets or returns the outer width of a window, including toolbars/scrollbars
pageXOffset	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
pageYOffset	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
parent	Returns the parent window of the current window
screen	Returns the Screen object for the window (See Screen object)
screenLeft	Returns the x coordinate of the window relative to the screen
screenTop	Returns the y coordinate of the window relative to the screen
screen	Returns the x coordinate of the window relative to the screen
screen	Returns the y coordinate of the window relative to the screen
self	Returns the current window

status	Sets the text in the statusbar of a window
top	Returns the topmost browser window

DOM(DocumentObjectModel)“Window”ObjectMethods

Method	Description
alert()	Displays an alert box with a message and an OK button
blur()	Removes focus from the current window
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
createPopup()	Creates a pop-up window
focus()	Sets focus to the current window
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	
scrollBy()	Scrolls the content by the specified number of pixels

scrollTo()	Scrolls the content to the specified coordinates
setInterval()	Calls a function or evaluates an expression at specified intervals (in milliseconds)
setTimeout()	Calls a function or evaluates an expression after a specified number of milliseconds

DOM(DocumentObjectModel)“Document”Object

Each HTML document loaded into a browser window becomes a Document object. The Document object provides access to all HTML elements in a page, from within a script. The Document object is also part of the Window object, and can be accessed through the *window.document* property.

DOM(DocumentObjectModel)“Document”ObjectProperties

Property	Description
anchors	Returns a collection of all the anchors in the document
applets	Returns a collection of all the applets in the document
body	Returns the body element of the document
cookie	Returns all name/value pairs of cookies in the document
documentMode	Returns the mode used by the browser to render the document
domain	Returns the domain name of the server that loaded the document
forms	Returns a collection of all the forms in the document
images	Returns a collection of all the images in the document
lastModified	Returns the date and time the document was last modified
links	Returns a collection of all the links in the document
readyState	Returns the (loading) status of the document
referrer	Returns the URL of the document that loaded the current document
title	Sets or returns the title of the document

URL	Returns the full URL of the document
-----	--------------------------------------

DOM(DocumentObjectModel)"Document"ObjectMethods

Method	Description
close()	Closes the output stream previously opened with document.open()
getElementsByName()	Accesses all elements with a specified name
write()	Writes HTML expressions or JavaScript code to a document
writeln()	Same as write(), but adds a newline character after each statement

DOM(DocumentObjectModel)"Navigator"Object

The navigator object contains information about the browser from where user is accessing your page.

DOM(DocumentObjectModel)"Navigator"ObjectProperties

Property	Description
appCodeName	Returns the code name of the browser
appName	Returns the name of the browser
appVersion	Returns the version information of the browser
cookieEnabled	Determines whether cookies are enabled in the browser
onLine	Boolean, returns <i>true</i> if the browser is on line, otherwise <i>false</i> .
platform	Returns for which platform the browser is compiled
userAgent	Returns the user-agent header sent by the browser to the server

DOM(DocumentObjectModel)"History"Object

The history object contains the URLs visited by the user (within a browser window). The history object is part of the window object and is accessed through the *window.history* property.

DOM(DocumentObjectModel)"History"ObjectProperties

Property	Description
length	Returns the number of URLs in the history list

DOM(DocumentObjectModel)"History"ObjectMethods

Method	Description
back()	Loads the previous URL in the history list
forward()	Loads the next URL in the history list
go()	Loads a specific URL from the history list

Example: The following example demonstrates how to use *open()* method of *window* object and *opener* property of *window* object and *document* object's *bgcolor* property. First open parent.html in browser that will automatically open child.html in a new window or tab and then click on buttons in child.html to change background color of parent.html.

Parent.Html

```
<html>
<head>
<script
type="text/javascript">var ch=window.open("child.html","win1","width=300,height=300
"); ch.document.bgColor="purple";
</script>
</head>
</html>
Child.Html
<html>
<head>
```

```

<title>Secondary window</title>
</head>
<body>
<center>
  <form>
    <input type="button" onClick=
      "window.opener.document.bgColor='yellow'" value="yellow">&nbsp;

    <input type="button" onClick=
      "window.opener.document.bgColor='lightgreen'" value="lightgreen">&nbsp;
    <input type="button" onClick=
      "window.opener.document.bgColor='white'" value="white">
  </form>
</center>
</body>
</html>

```

Example: The following example demonstrates various properties of window object that can be specified using third argument of *open* method of *window* object.

```

<html>
<head>
<script type="text/javascript"> function newwindow()

{
var          win=window.open("password.html", "Win1",          "width=300,height=300,resizable,
toolbar,left=100,top=100");
}
</script>
</head>
<body onload="newwindow()">
</body>
</html>

```

Example : The following example demonstrates the properties of *navigator* object

```

<html>
<body>
<script type="text/javascript">document.write("<h3> Your Browser is : " + navigator.appName +
"</h3>");

document.write("<h3> Your Browser Version is : " + navigator.appVersion +
"</h3>"); document.write("<h3> Your Platform is : " + navigator.platform + "</h3>"); </script>

</body>
</html>

```

Example : The following example demonstrates the use *setInterval()* and *clearInterval()* methods of *window* object.

```
<html>
<head>
<script      type="text/javascript">var id,i=0;      var      colors=new
Array("red","green","blue","yellow","white"); function FStart()
{
id=setInterval("ChangeColor()",1000);
}
function FStop()
{
clearInterval(id);
}
function ChangeColor()
{
document.bgColor=colors[i];
i++; if(i==5)

i=0;
}
</script>
</head>
<body>
<input type="button" value="Start" onclick="FStart()"/>
<input type="button" value="Stop" onclick="FStop()"/>
</body>
</html>
```

Example : The following example demonstrates the use of *history* object.

First.Html

```
<!doctype html>
<html>
<body>
<a href="history.html">
</a>
</body>
</html>
```

History.Html

```
<!doctype html>
<html>
<body>
```

```
<!doctype html>
<html>
<body>


</body>
</html>
```

www.enosislearning.com