HTML5

HTML5 is the latest and most enhanced version of HTML.Technically, HTML is not a programming language, but rather a mark up language.

The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plug-ins such as Adobe Flash, Microsoft Silverlight, and Google Gears.HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web.

New Features

HTML5 introduces a number of new elements and attributes that helps in building a modern website. Following are great features introduced in HTML5.

- **New Semantic Elements** − These are like <header>, <footer>, and <section>.
- **Forms 2.0** − Improvements to HTML web forms where new attributes have been introduced for <input> tag.
- **Persistent Local Storage** − To achieve without resorting to third-party plugins.
- **WebSocket** − A a next-generation bidirectional communication technology for web applications.
- **Server-Sent Events** − HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
- **Canvas** − This supports a two-dimensional drawing surface that you can program with JavaScript.
- **Audio & Video** − You can embed audio or video on your web pages without resorting to third-party plugins.
- **Geolocation** − Now visitors can choose to share their physical location with your web application.
- **Microdata** − This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
- **Drag and drop** − Drag and drop the items from one location to another location on a the same webpage.

The HTML 5 language has a "custom" HTML syntax that is compatible with HTML 4 and XHTML1 documents published on the Web, but is not compatible with the more esoteric SGML features of HTML 4.

HTML 5 does not have the same syntax rules as XHTML where we needed lower case tag names, quoting our attributes,an attribute had to have a value and to close all empty elements.

But HTML5 is coming with lots of flexibility and would support the followings −

- Uppercase tag names.
- Quotes are optional for attributes.
- Attribute values are optional.
- Closing empty elements are optional.

The DOCTYPE

DOCTYPEs in older versions of HTML were longer because the HTML language was SGML based and therefore required a reference to a DTD.

HTML 5 authors would use simple syntax to specify DOCTYPE as follows −

```
<!DOCTYPE html>
```

All the above syntax is case-insensitive.

Character Encoding

HTML 5 authors can use simple syntax to specify Character Encoding as follows −

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

All the above syntax is case-insensitive.

The <script> tag

It's common practice to add a type attribute with a value of "text/javascript" to script elements as follows −

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

HTML 5 removes extra information required and you can use simply following syntax −

```
<script src="scriptfile.js"></script>
```

The <link> tag

So far you were writing <link> as follows −

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

HTML 5 removes extra information required and you can use simply following syntax −

```
<link rel="stylesheet" href="stylefile.css">
```

HTML5 Elements

HTML5 elements are marked up using start tags and end tags. Tags are delimited using angle brackets with the tag name in between.

The difference between start tags and end tags is that the latter includes a slash before the tag name.

Following is the example of an HTML5 element −

```
<p>...</p>
```

HTML5 tag names are case insensitive and may be written in all uppercase or mixed case, although the most common convention is to stick with lower case.

Most of the elements contain some content like ... contains a paragraph. Some elements, however, are forbidden from containing any content at all and these are known as void elements. For example, br, hr, link and meta etc.

Tag	Description
<!--...-->	Specifies a comment
<!DOCTYPE>	Specifies the document type
<a>	Specifies an anchor
<audio>	New Tag:Specifies an audio file.
<base>	Specifies a base URL for all the links in a page
<basefont>	Deprecated: Specifies a base font
<button>	Specifies a push button
<canvas>	New Tag:This is used for rendering dynamic bitmap graphics on the fly, such as graphs or games.
<footer>	New Tag:Specifies a footer for a section and can contain information about the author, copyright information, et cetera.
<header>	New Tag:Specifies a group of introductory or navigational aids.
<hgroup>	New Tag:Specifies the header of a section.
<marquee>	Create a scrolling-text marquee
<menu>	Deprecated: Specifies a menu list
<meter>	New Tag:Specifies a measurement, such as disk usage.
<multicol>	Specifies a multicolumn text flow
<nav>	New Tag:Specifies a section of the document intended for navigation.

HTML5 Attributes

Elements may contain attributes that are used to set various properties of an element.

Some attributes are defined globally and can be used on any element, while others are defined for specific elements only. All attributes have a name and a value and look like as shown below in the example.

Following is the example of an HTML5 attributes which illustrates how to mark up a div element with an attribute named class using a value of "example" –

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

Attributes may only be specified within start tags and must never be used in end tags.

HTML5 attributes are case insensitive and may be written in all upper case or mixed case, although the most common convention is to stick with lower case.

Attribute	Options	Function
accesskey	User Defined	Specifies a keyboard shortcut to access an element.
align	right, left, center	Horizontally aligns tags
background	URL	Places an background image behind an element
bgcolor	numeric, hexidecimal, RGB values	Places a background color behind an element
class	User Defined	Classifies an element for use with Cascading Style Sheets.
contenteditable	true, false	Specifies if the user can edit the element's content or not.
contextmenu	Menu id	Specifies the context menu for an element.
data-XXXX	User Defined	Custom attributes. Authors of a HTML document can define their own attributes. Must start with "data-".
draggable	true,false, auto	Specifies whether or not a user is allowed to drag an element.
height	Numeric Value	Specifies the height of tables, images, or table cells.
hidden	hidden	Specifies whether element should be visible or not.
Id	User Defined	Names an element for use with Cascading Style Sheets.
item	List of elements	Used to group elements.
itemprop	List of items	Used to group items.
spellcheck	true, false	Specifies if the element must have it's spelling or grammar checked.
style	CSS Style sheet	Specifies an inline style for an element.
subject	User define id	Specifies the element's corresponding item.

tabindex	Tab number	Specifies the tab order of an element.
title	User Defined	"Pop-up" title for your elements.
valign	top, middle, bottom	Vertically aligns tags within an HTML element.
width	Numeric Value	Specifies the width of tables, images, or table cells.

For a complete list of HTML5 Tags and related attributes please check reference to HTML5 Tags.

Custom Attributes

A new feature being introduced in HTML 5 is the addition of custom data attributes.

A custom data attribute starts with **data-** and would be named based on your requirement. Following is the simple example –

```
<div class="example" data-subject="physics" data-level="complex">

  ...

</div>
```

The above will be perfectly valid HTML5 with two custom attributes called *data-subject* and *data-level*. You would be able to get the values of these attributes using JavaScript APIs or CSS in similar way as you get for standard attributes.

HTML5 Document

The following tags have been introduced for better structure –

- **section** – This tag represents a generic document or application section. It can be used together with h1-h6 to indicate the document structure.
- **article** – This tag represents an independent piece of content of a document, such as a blog entry or newspaper article.
- **aside** – This tag represents a piece of content that is only slightly related to the rest of the page.
- **header** – This tag represents the header of a section.
- **footer** – This tag represents a footer for a section and can contain information about the author, copyright information, etc.
- **nav** – This tag represents a section of the document intended for navigation.
- **dialog** – This tag can be used to mark up a conversation.
- **figure** – This tag can be used to associate a caption together with some embedded content, such as a graphic or video.

The markup for an HTM 5 document would look like the following –

```
<!DOCTYPE html>
```

```
<html>


   <head>
      <meta charset="utf-8">
      <title>...</title>
   </head>


   <body>
      <header>...</header>
      <nav>...</nav>


      <article>
         <section>
            ...
         </section>
      </article>


      <aside>...</aside>
      <figure>...</figure>
      <footer>...</footer>


   </body>
</html>
```

```
<!DOCTYPE html>


<html>


   <head>
      <meta charset="utf-8">
      <title>...</title>
```

```
</head>

<body>

   <header role="banner">
      <h1>HTML5 Document Structure Example</h1>
      <p>This page should be tried in safari, chrome or Mozila.</p>
   </header>

   <nav>

      <ul>
         <li><a href="#">HTML Tutorial</a></li>
         <li><a href="#">CSS Tutorial</a></li>
         <li><a href="#">JavaScript Tutorial</a></li>
      </ul>

   </nav>

   <article>
      <section>
         <p>Once article can have multiple sections</p>
      </section>
   </article>

   <aside>
      <p>This is  aside part of the web page</p>
   </aside>

   <figure align="right">
```

```
        <img src="/html5/images/logo.png" alt="TutorialPoint" width="200" height="100">
      </figure>

      <footer>
         <p>Created by <a href="#">Tutorials Point</a></p>
      </footer>

   </body>
</html>
```

Output:

HTML5 Document Structure Example

This page should be tried in safari, chrome or Mozila.

- HTML Tutorial
- CSS Tutorial
- JavaScript Tutorial

Once article can have multiple sections

This is aside part of the web page

HTML5 - Web Forms 2.0

Web Forms 2.0 is an extension to the forms features found in HTML4. Form elements and attributes in HTML5 provide a greater degree of semantic mark-up than HTML4 and remove a great deal of the need for tedious scripting and styling that was required in HTML4.

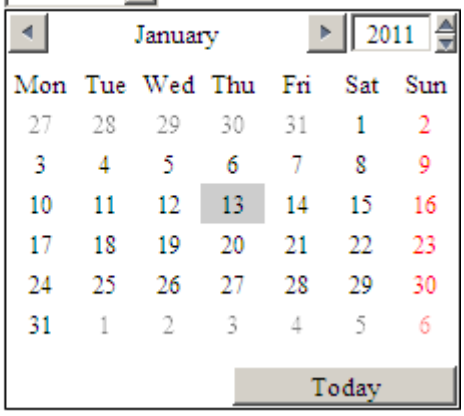
HTML5 input elements introduced several new values for the **type** attribute

A Date and time field can be easily found in many web forms. Typical applications are like ticket booking, appointment booking, ordering pizza and etc.

The most commonly used solution for date input is to use Javascript date picker. As of writing, the only web browser completely support date time input is Opera (v11) and Google Chrome (v20). In HTML5, it is the job of web browser to ensure user can only enter a valid date time string into the input textbox.

If you do not have the web browser that support it, the picture below shows you how it looks like.

Meeting Date : 2011-01-13



With HTML5 support, web designer no longer needs to download fancy Javascript control for basic date input. Exactly what you need to do is just <input type="date"/>

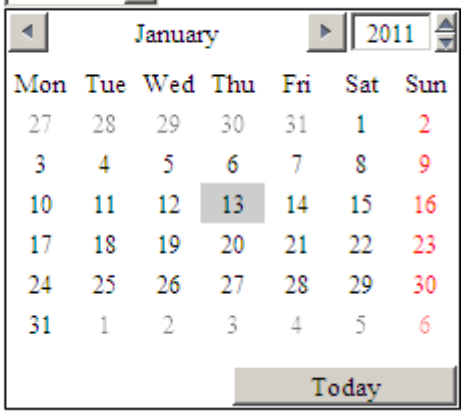
```
<label for="meeting">Meeting Date : </label><input id="meeting" type="date" value="2011-01-13"/>
```

Once again HTML5 has made our life easier! Furthermore, you are not just getting "Date" input, there are half a dozen of "Date Time" related inputs that you can pick and use! Of course, the ball is now in the court of the web browser companies as they decide when to implement this standard.

1. Date (<input type="date"/>)

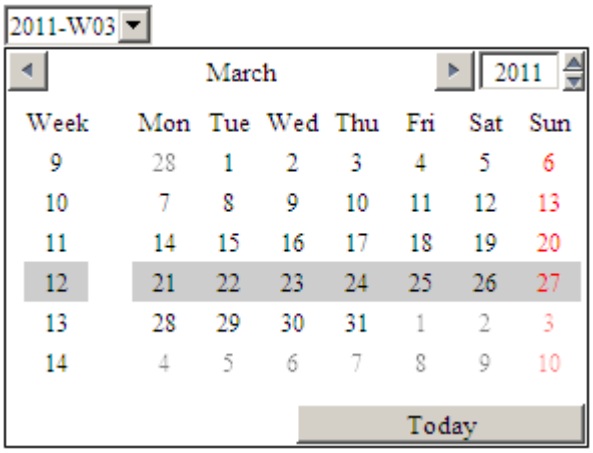
Apparently this is browser native date picker. You can only pick a specific date from the calendar.

Meeting Date : 2011-01-13



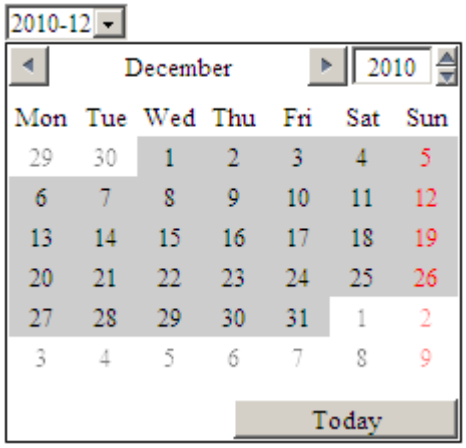
2. Week (<input type="week"/>)

Instead of picking a date, you can pick a week too. Please notice the Week number on the left of the calendar.



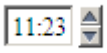
3. Month (<input type="month"/>)

You can even have a month picker, here the calendar that allows you to choose a month in a year.



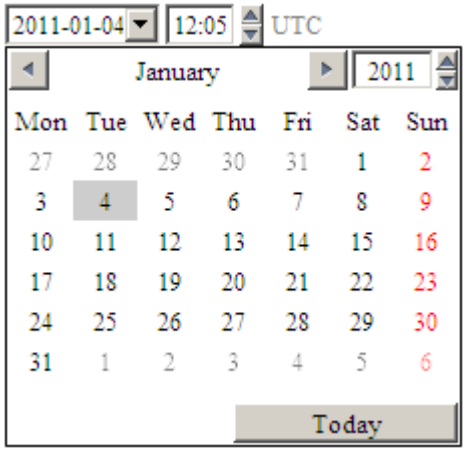
4. Time (<input type="time"/>)

This is nothing special, a time picker for time input.



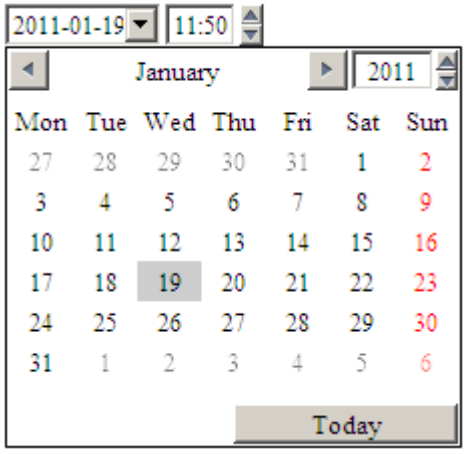
5. Date and Time (<input type="datetime"/>)

You can choose date and time with time zone. Input value is represented in UTC time.



6. Local Date and Time (<input type="datetime-local"/>)

In compared to UTC time, you can have input date time value represents local time too.



Let's look into the related attributes that we possibly use.

Attributes	Descriptions
value	Value is the default value of the input box when a page is first loaded. This is a common attribute for <input> element regardless which type you are using.
min	Minimum Date or time
max	Maximum Date or time
step	Step scale factor. Different type of input has its own default "step" value. • Date - default is 1 day • Week - default is 1 week • Month - default is 1 month • Time - default is 1 minute • DateTime - default is 1 minute • Local DateTime - default is also 1 minute

HTML5 – Canvas

HTML5 element <canvas> gives you an easy and powerful way to draw graphics using JavaScript. It can be used to draw graphs, make photo compositions or do simple (and not so simple) animations.

Here is a simple <canvas> element which has only two specific attributes**width** and **height** plus all the core HTML5 attributes like id, name and class etc.

```
<canvas id="mycanvas" width="100" height="100"></canvas>
```

You can easily find that <canvas> element in the DOM using *getElementById()*method as follows –

```
var canvas  = document.getElementById("mycanvas");
```

Let us see a simple example on using <canvas> element in HTML5 document.

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

      <style>
         #mycanvas{border:1px solid red;}
      </style>

   </head>
   <body>

      <canvas id="mycanvas" width="100" height="100"></canvas>

   </body>
</html>
```

Drawing with Java Script.

<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

ctx.fillStyle = "#FF0000";

ctx.fillRect(0,0,150,75);

</script>

Draw a line

```
<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

ctx.moveTo(0,0);

ctx.lineTo(200,100);

ctx.stroke();

</script>
```

Draw a Circle

```
<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

ctx.beginPath();

ctx.arc(95,50,40,0,2*Math.PI);

ctx.stroke();

</script>
```

Audio & Video

HTML5 features, include native audio and video support without the need for Flash.

The HTML5 <audio> and <video> tags make it simple to add media to a website. You need to set **src** attribute to identify the media source and include a controls attribute so the user can play and pause the media.

Embedding Video

Here is the simplest form of embedding a video file in your webpage −

```
<video src="foo.mp4"  width="300" height="200" controls>
  Your browser does not support the <video> element.
</video>
```

The current HTML5 draft specification does not specify which video formats browsers should support in the video tag. But most commonly used video formats are −

- **Ogg** − Ogg files with Thedora video codec and Vorbis audio codec.

- **mpeg4** – MPEG4 files with H.264 video codec and AAC audio codec.

You can use <source> tag to specify media along with media type and many other attributes. A video element allows multiple source elements and browser will use the first recognized format –

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

    <video  width="300" height="200" controls autoplay>
      <source src="/html5/foo.ogg" type="video/ogg" />
      <source src="/html5/foo.mp4" type="video/mp4" />
      Your browser does not support the video element.
    </video>

  </body>
</html>
```

Embedding Audio

HTML5 supports <audio> tag which is used to embed sound content in an HTML or XHTML document as follows.

```
<audio src="foo.wav" controls autoplay>
  Your browser does not support the <audio> element.
</audio>
```

The current HTML5 draft specification does not specify which audio formats browsers should support in the audio tag. But most commonly used audio formats are **ogg, mp3** and **wav**.

You can use <source> tag to specify media along with media type and many other attributes. An audio element allows multiple source elements and browser will use the first recognized format –

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

    <audio controls autoplay>
      <source src="/html5/audio.ogg" type="audio/ogg" />
      <source src="/html5/audio.wav" type="audio/wav" />
      Your browser does not support the audio element.
    </audio>
```

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

CSS3 Modules

CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added.

Some of the most important CSS3 modules are:

- Selectors
- Box Model
- Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

Most of the new CSS3 properties are implemented in modern browsers.

Rounded Corners

CSS3 Rounded corners are used to add special colored corner to body or text by using the border-radius property.A simple syntax of rounded corners is as follows.

Values	Description
border-radius	Use this element for setting four boarder radius property
border-top-left-radius	Use this element for setting the boarder of top left corner
border-top-right-radius	Use this element for setting the boarder of top right corner
border-bottom-right-radius	Use this element for setting the boarder of bottom right corner
border-bottom-left-radius	Use this element for setting the boarder of bottom left corner

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

```
    <style>
      #rcorners1 {
        border-radius: 25px;
        background: #8AC007;
        padding: 20px;
        width: 200px;
        height: 150px;
      }
      #rcorners2 {
        border-radius: 25px;
        border: 2px solid #8AC007;
        padding: 20px;
        width: 200px;
        height: 150px;
      }
    </style>

  </head>
  <body>
    <p id="rcorners1">Rounded corners!</p>
    <p id="rcorners2">Rounded corners!</p>
    <p id="rcorners3">Rounded corners!</p>
  </body>
</html>
```

OUTPUT:

Multi Background

CSS Multi background property is used to add one or more images at a time without HTML code, We can add images as per our requirement.A sample syntax of multi background images is as follows –

Values	Description
background	Used to setting all the background image properties in one section
background-clip	Used to declare the painting area of the background
background-image	Used to specify the background image
background-origin	Used to specify position of the background images
background-size	Used to specify size of the background images

Following is the example which demonstrate the multi background images

```
<html>
  <head>

    <style>
      #multibackground {
        background-image: url(/css/images/logo.png), url(/css/images/border.png);
        background-position: left top, left top;
        background-repeat: no-repeat, repeat;
        padding: 75px;
      }
    </style>

  </head>
  <body>

    <div id="multibackground">
      <h1>www.tutorialspoint.com</h1>

<p>
```

```
The journey commenced with a single tutorial on HTML in  2006 and elated by the response it generated,
      we worked our way to adding fresh tutorials to our repository which now proudly flaunts a wealth of tutorials and allied articles on
      topics ranging from programming languages to web designing to academics and much more..</p>
    </div>


  </body>
</html>
```

Multi background property is accepted to add different sizes for different images.A sample syntax is as shown below –

```
#multibackground {
  background: url(/css/imalges/logo.png) left top no-repeat, url(/css/images/boarder.png) right bottom no-repeat, url(/css/images/css.gif) left top repeat;
  background-size: 50px, 130px, auto;
}
```

shadow

CSS3 supported to add shadow to text or elements.Shadow property has divided as follows

- Text shadow
- Box Shadow

Text shadow

CSS3 supported to add shadow effects to text. Following is the example to add shadow effects to text

```
<html>
  <head>


    <style>
      h1 {
        text-shadow: 2px 2px;
      }
      h2 {
        text-shadow: 2px 2px red;
      }
      h3 {
        text-shadow: 2px 2px 5px red;
      }
```

```
    h4 {
      color: white;
      text-shadow: 2px 2px 4px #000000;
    }
    h5 {
      text-shadow: 0 0 3px #FF0000;
    }
    h6 {
      text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
    }
    p {
      color: white;
      text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;
    }
  </style>

</head>
<body>

  <h1>HTML Webpage</h1>
  <h2> HTML Webpage </h2>
  <h3> HTML Webpage </h3>
  <h4> HTML Webpage </h4>
  <h5> HTML Webpage </h5>
  <h6> HTML Webpage </h6>
  <p> HTML Webpage /p>

</body>
</html>
```

Output:

2d Transforms

2D transforms are used to re-change the element structure as translate, rotate, scale, and skew

The following table has contained common values which are used in 2D transforms

Values	Description
matrix(n,n,n,n,n,n)	Used to defines matrix transforms with six values
translate(x,y)	Used to transforms the element along with x-axis and y-axis
translateX(n)	Used to transforms the element along with x-axis
translateY(n)	Used to transforms the element along with y-axis
scale(x,y)	Used to change the width and height of element
scaleX(n)	Used to change the width of element
scaleY(n)	Used to change the height of element
rotate(angle)	Used to rotate the element based on an angle
skewX(angle)	Used to defines skew transforms along with x axis
skewY(angle)	Used to defines skew transforms along with y axis

```
<html>
  <head>

    <style>
```

```
      div {
        width: 300px;
        height: 100px;
        background-color: pink;
        border: 1px solid black;
      }
      div#myDiv {
        /* IE 9 */
        -ms-transform: rotate(20deg);

        /* Safari */
        -webkit-transform: rotate(20deg);

        /* Standard syntax */
        transform: rotate(20deg);
      }
    </style>

  </head>
  <body>
    <div>
    Tutorials point.com.
    </div>

    <div id="myDiv">
    Tutorials point.com
    </div>
  </body>
</html>
```

3D transforms

Using with 3d transforms, we can move element to x-axis, y-axis and z-axis, Below example clearly specifies how the element will rotate

Methods of 3D transforms

Below methods are used to call 3D transforms

Values	Description
matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	Used to transforms the element by using 16 values of matrix
translate3d(x,y,z)	Used to transforms the element by using x-axis,y-axis and z-axis
scaleX(x)	Used to scale transforms the element by using x-axis
scaleY(y)	Used to scale transforms the element by using y-axis
scaleY(y)	Used to transforms the element by using z-axis
rotateX(angle)	Used to rotate transforms the element by using x-axis
rotateY(angle)	Used to rotate transforms the element by using y-axis
rotateZ(angle)	Used to rotate transforms the element by using z-axis

```
<html>
  <head>
    <style>
      div {
        width: 200px;
        height: 100px;
        background-color: pink;
        border: 1px solid black;
```

```
      }
      div#myDiv {
        -webkit-transform: rotateX(150deg);

        /* Safari */
        transform: rotateX(150deg);

        /* Standard syntax */
      }
    </style>

  </head>
  <body>

    <div>
    tutorials point.com
    </div>

    <p>Rotate X-axis</p>

    <div id="myDiv">
    tutorials point.com.
    </div>

  </body>
</html>
```

Box Sizing

Box sizing property is using to change the height and width of element.

width + padding + border = actual visible/rendered width of an element's box

height + padding + border = actual visible/rendered height of an element's box

```
<html>
  <head>

    <style>
      .div1 {
        width: 300px;
        height: 100px;
        border: 1px solid blue;
        box-sizing: border-box;
      }
      .div2 {
        width: 300px;
        height: 100px;
        padding: 50px;
        border: 1px solid red;
        box-sizing: border-box;
      }
    </style>

  </head>
  <body>
    <div class="div1">TutorialsPoint.com</div></br>
    <div class="div2">TutorialsPoint.com</div>
  </body>
</html>
```

Output:

@media Rule

The @media rule is used to define different style rules for different media types/devices.

In CSS2 this was called media types, while in CSS3 it is called media queries.

Media queries look at the capability of the device, and can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution.

Syntex:

<link rel="stylesheet" media="*mediatype* and|not|only (*media feature*)" href="*mystylesheet.css*">

Media Types

Value	Description
All	Used for all media type devices
Aural	Deprecated. Used for speech and sound synthesizers
Braille	Deprecated. Used for braille tactile feedback devices
embossed	Deprecated. Used for paged braille printers

handheld	Deprecated. Used for small or handheld devices
Print	Used for printers
projection	Deprecated. Used for projected presentations, like slides
Screen	Used for computer screens, tablets, smart-phones etc.
Speech	Used for screenreaders that "reads" the page out loud

Media Features

Value	**Description**
aspect-ratio	The ratio between the width and the height of the viewport
Color	The number of bits per color component for the output device
color-index	The number of colors the device can display
device-aspect-ratio	The ratio between the width and the height of the device
device-height	The height of the device, such as a computer screen
device-width	The width of the device, such as a computer screen
Grid	Whether the device is a grid or bitmap
Height	The viewport height

max-aspect-ratio	The maximum ratio between the width and the height of the display area
max-color	The maximum number of bits per color component for the output device

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}

@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
</style>
</head>
<body>

<h1>Resize the browser window to see the effect!</h1>
<p>The media query will only apply if the media type is screen and the viewport is 480px wide or wider.</p>

</body>
</html>
```