SCS1620 USER INTERFACE TECHNOLOGIES

Unit I

Syllabus

HTML5: What is HTML5 - Features of HTML5 – Semantic Tags – New Input Elements and tags - Media tags (audio and video tags) – Designing Graphics using Canvas API - Drag and Drop features – Geolocation API - Web storage (Session and local storage).

CSS3: What is CSS3 – Features of CSS3 – Implementation of border radius, box shadow, image border, custom web font, backgrounds - Advanced text effects(shadow) - 2D and 3D Transformations - Transitions to elements - Animations to text and elements

HyperText Markup Language (HTML)

HTML5 is the latest evolution of the standard that defines HTML.

The term represents two different concepts.

- New version of the language HTML, with new elements, attributes, and behaviors
- Larger set of technologies that allows the building of more diverse and powerful Web sites and applications.

This set is sometimes called HTML5 & friends and often shortened to just HTML5.

Using the Version 5:

The DOCTYPE declaration for HTML5:

```
<!DOCTYPE html>
```

Character encoding (Character Set):

```
<meta charset=”UTF-8”>
```

First Program:

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

New HTML5 Elements

Semantic elements	<header>, <footer>, <article>, and <section>
Attributes of form elements	number, date, time, calendar, and range
Graphic elements	<svg> (Scalable Vector Graphics) and <canvas> (using java script – on the fly)
Multimedia elements	<audio>, <video>, <embed>, <track>, <source>

API's in HTML5

- HTML Geolocation
- HTML Drag and Drop
- HTML Local Storage
- HTML Application Cache
- HTML Web Workers (Java Script runs in back ground)
- HTML SSE (Server Sent Events)

HTML History

Year	Version
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	W3C Recommendation: HTML 3.2##
1999	W3C Recommendation: HTML 4.01##
2000	W3C Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft**
2012	WHATWG HTML5 Living Standard**
2014	W3C Recommendation: HTML5
2016	W3C Candidate Recommendation: HTML 5.1

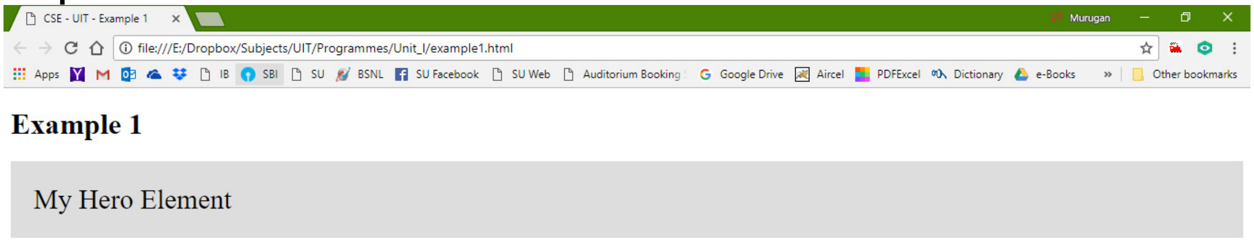
**Web Hypertext Application Technology Working Group (WHATWG) (https://whatwg.org/)
##World Wide Web Consortium (W3C) (https://www.w3.org/)

- From 1991 to 1999, HTML developed from version 1 to version 4.
- In year 2000, the World Wide Web Consortium (W3C) recommended XHTML 1.0. The XHTML syntax was strict, and the developers were forced to write valid and "well-formed" code.
- In 2004, W3C's decided to close down the development of HTML, in favor of XHTML.
- In 2004, WHATWG (Web Hypertext Application Technology Working Group) was formed. The WHATWG wanted to develop HTML, consistent with how the web was used, while being backward compatible with older versions of HTML.
- In 2004 - 2006, the WHATWG gained support by the major browser vendors.
- In 2006, W3C announced that they would support WHATWG.
- In 2008, the first HTML5 public draft was released.
- In 2012, WHATWG and W3C decided on a separation:
- WHATWG wanted to develop HTML as a "Living Standard". A living standard is always updated and improved. New features can be added, but old functionality cannot be removed.
- The WHATWG HTML5 Living Standard was published in 2012, and is continuously updated.
- W3C wanted to develop a definitive HTML5 and XHTML standard.
- The W3C HTML5 recommendation was released 28 October 2014.
- W3C also published an HTML 5.1 Candidate Recommendation on 21 June 2016.

Example 1

```
<!DOCTYPE html>
<html>
<head>
  <title>CSE - UIT - Example 1</title>
  <style>
         testElement {
      display: block;
      background-color: #dddddd;
      padding: 50px;
      font-size: 30px;
         }
  </style>
</head>
<body>
         <h1>Example 1</h1>
         <testElement>My Hero Element</testElement>
</body>
</html>
```

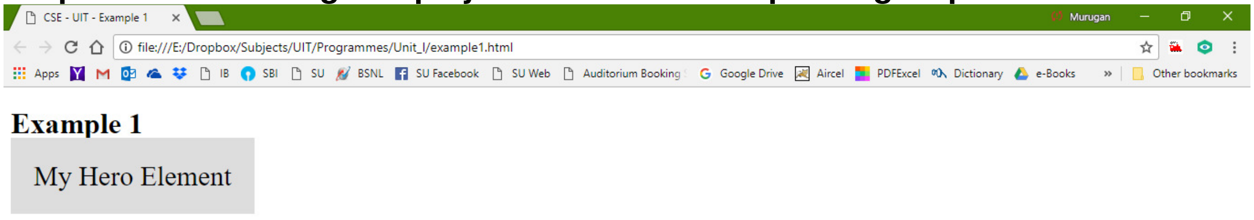
Output:



Example 1

My Hero Element

Output after removing - display: block and set the padding: 25px



Example 1

My Hero Element

Semantic/Structural Elements

Tag	**Description**
<article>	Defines an article in a document
<aside>	Defines content aside from the page content
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<figcaption>	Defines a caption for a <figure> element
<figure>	Defines self-contained content
<footer>	Defines a footer for a document or section
<header>	Defines a header for a document or section
<main>	Defines the main content of a document

<mark>	Defines marked/highlighted text
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links
<progress>	Represents the progress of a task
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time
<wbr>	Defines a possible line-break

Form Elements

Tag	Description
<datalist>	Specifies a list of pre-defined options for input controls
<output>	Defines the result of a calculation

Input Types

New Input Types	New Input Attributes
color	autocomplete
date	autofocus
datetime	form
datetime-local	formaction
email	formenctype
month	formmethod
number	formnovalidate
range	formtarget
search	height and width
tel	list
time	min and max
url	multiple
week	pattern (regexp)
	placeholder
	required
	step

HTML 5 Semantic Elements

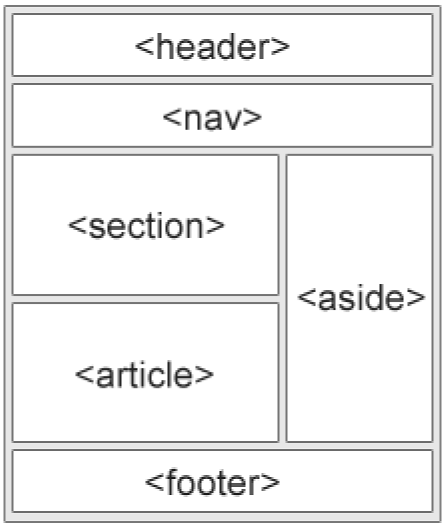
- Semantics is the study of the meanings of words and phrases in a language
- Semantic elements = elements with a meaning
- A semantic element clearly describes its meaning to both the browser and the developer
- Examples of non-semantic elements:
    - <div> and <span> - Tells nothing about its content
- Examples of semantic elements:
    - <form>, <table>, and <article> - Clearly defines its content

Many web sites contain HTML code like:

```
<div id="nav">              (indicates navigation)
<div class="header">        (indicates header)
<div id="footer">           (indicates footer)
```

HTML5 offers new semantic elements to define different parts of a web page:

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



<section> Element

According to W3C's HTML5 documentation:

"A section is a thematic grouping of content, typically with a heading."

A home page could normally be split into sections for introduction, content, and contact information.

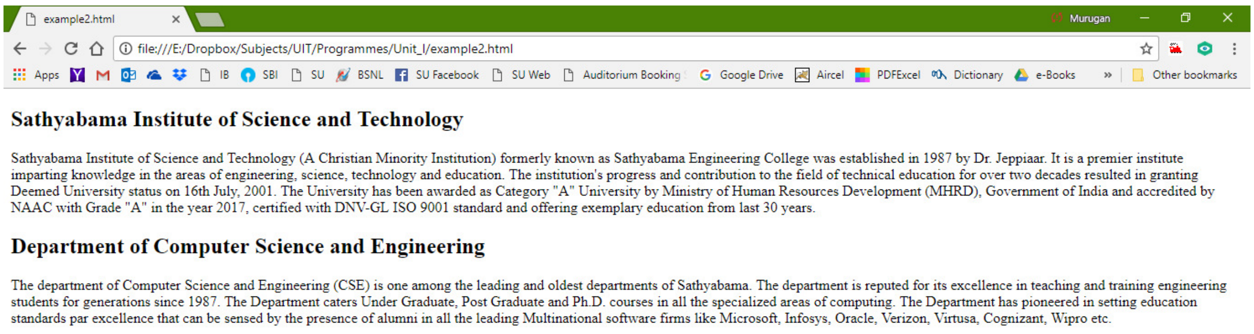
Example 2

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

<section>
  <h1>Sathyabama Institute of Science and Technology</h1>
  <p>Sathyabama Institute of Science and Technology (A Christian Minority Institution) formerly known as Sathyabama Engineering College was established in 1987 by Dr. Jeppiaar. It is a premier institute imparting knowledge in the areas of engineering, science, technology and education. The institution's progress and contribution to the field of technical education for over two decades resulted in granting Deemed University status on 16th July, 2001. The University has been awarded as Category "A" University by Ministry of Human Resources Development (MHRD), Government of India and accredited by NAAC with Grade "A" in the year 2017, certified with DNV-GL ISO 9001 standard and offering exemplary education from last 30 years.</p>
</section>

<section>
  <h1>Department of Computer Science and Engineering</h1>
  <p>The department of Computer Science and Engineering (CSE) is one among the leading and oldest departments of Sathyabama. The department is reputed for its excellence in teaching and training engineering students for generations since 1987. The Department caters Under Graduate, Post Graduate and Ph.D. courses in all the specialized areas of
```

```
computing.  The Department has pioneered in setting education standards par excellence that can be sensed by the presence of alumni in all the leading Multinational software firms like Microsoft, Infosys, Oracle, Verizon, Virtusa, Cognizant, Wipro etc.</p>
</section>
</body>
</html>
```


example2.html
Murugan
file:///E:/Dropbox/Subjects/UIT/Programmes/Unit_I/example2.html
Apps IB SBI SU BSNL SU Facebook SU Web Auditorium Booking Google Drive Aircel PDFExcel Dictionary e-Books Other bookmarks

Sathyabama Institute of Science and Technology

Sathyabama Institute of Science and Technology (A Christian Minority Institution) formerly known as Sathyabama Engineering College was established in 1987 by Dr. Jeppiaar. It is a premier institute imparting knowledge in the areas of engineering, science, technology and education. The institution's progress and contribution to the field of technical education for over two decades resulted in granting Deemed University status on 16th July, 2001. The University has been awarded as Category "A" University by Ministry of Human Resources Development (MHRD), Government of India and accredited by NAAC with Grade "A" in the year 2017, certified with DNV-GL ISO 9001 standard and offering exemplary education from last 30 years.

Department of Computer Science and Engineering

The department of Computer Science and Engineering (CSE) is one among the leading and oldest departments of Sathyabama. The department is reputed for its excellence in teaching and training engineering students for generations since 1987. The Department caters Under Graduate, Post Graduate and Ph.D. courses in all the specialized areas of computing. The Department has pioneered in setting education standards par excellence that can be sensed by the presence of alumni in all the leading Multinational software firms like Microsoft, Infosys, Oracle, Verizon, Virtusa, Cognizant, Wipro etc.

<header> Element

The <header> element specifies a header for a document or section.
The <header> element should be used as a container for introductory content.
Can have several <header> elements in one document.

```
<!DOCTYPE html>
<html>
<body>
<article>
  <header>
    <h1>Header</h1>
    <p>Paragraph</p>
  </header>
  <p>Some Text Goes Here</p>
</article>
</body>
</html>
```

<footer> Element

The <footer> element specifies the footer for a document or section.
A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.
Can have several <footer> elements in one document.

```
<!DOCTYPE html>
<html>
<body>
          <footer style="color: green;">Copyright &copy; (2018) SIST(DU), Chennai.</footer>
</body>
</html>
```

<nav> Element

The <nav> element defines a set of navigation links.

```
<!DOCTYPE html>
<html>
<body>
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
</body>
</html>
```

Output:

HTML | CSS | JavaScript | jQuery

<aside> Element

```
<!DOCTYPE html>
<html>
<body>
<p>Department of Computer Science and Engineering</p>
<aside>
  <h4>Elective Subjects</h4>
  <p>Internet of Things</p>
  <p>Python Programming</p>
  <p>User Interface Technologies</p>
</aside>
</body>
</html>
```

<figure> and <figcaption> Elements

In HTML5, an image and a caption can be grouped together in a <figure> element. The purpose of a figure caption is to add a visual explanation to an image.

```
<!DOCTYPE html>
<html>
<body>
<p>Browsers Logo</p>
<figure>
  <img src="chrome.jpg" alt="chrome" width="304" height="228">
  <figcaption>Fig.1 - Google Chrome</figcaption>
</figure>
<figure>
  <img src="firefox.jpg" alt="firefox" width="304" height="228">
  <figcaption>Fig.2 - Mozilla Firefox</figcaption>
</figure>
<figure>
  <img src="safari.jpg" alt="safari" width="304" height="228">
  <figcaption>Fig.3 - Apple Safari</figcaption>
</figure>
<figure>
  <img src="opera.jpg" alt="Opera" width="304" height="228">
  <figcaption>Fig.4 - Opera</figcaption>
</figure>
```

```
<figure>
  <img src="ie.jpg" alt="ie" width="304" height="228">
  <figcaption>Fig.5 - Microsoft Internet Explorer</figcaption>
</figure>
</body>
</html>
```

Input Types

a. Text, Password, Submit and Reset

```
<!DOCTYPE html>
<html>
<body>
<form action="/action_page.php">
        User name:<br><input type="text" name="userid"><br>
        User password:<br><input type="password" name="psw">
        <input type="submit" value="Login">
        <input type=”reset”>
</form>
<p>The characters in a password field are masked (shown as asterisks or circles).</p>
</body>
</html>
```

b. Radio & Check Box

```
<!DOCTYPE html>
<html>
<body>
<form action="/action_page.php">
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other<br><br>
  <input type="checkbox" name="course" value="NPTEL">NPTEL Course<br>
  <input type="checkbox" name="course" value="GIAN">GIAN Course
  <input type="submit">
</form>
</body>
</html>
```

c. Button

```
<!DOCTYPE html>
<html>
        <head><title>Test</title></head>
<body>
        <input type="button" onclick="alert('Hello World!')" value="Click Me!">
</body>
</html>
```

d. Color, date, datetime-local, email, month, numeric, range, tel, time, url

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

```html
<p>Depending on browser support:<br></p>
<form action="/action_page.php">
        <p>A color picker can pop-up when you enter the input field. Used in input fields that
should contain a color.</p>
        Select your favorite color:  <input type="color" name="favcolor" value="#0000ff">
        <p>A date picker can pop-up when you enter the input field.  Is used for input fields
that should contain a date.</p>
        Birthday: <input type="date" name="bday">
        <p>DatePicker with Restrictions</p>
        Enter a date before 01-01-1980:
        <input type="date" name="bday" max="1979-12-31"><br>
        Enter a date after 01-01-2000:
        <input type="date" name="bday" min="2000-01-02"><br>
        <p>Specifies a date and time input field, with no time zone.</p>
        Birthday (date and time):    <input type="datetime-local" name="bdaytime">
        <p>Selects Month & Year</p>
        Birthday (month and year): <input type="month" name="bdaymonth">
        <p>Allows the user to select a week and year.</p>
        Select a week: <input type="week" name="year_week">
        <p>email: Used for input fields that should contain an e-mail address. Depending on
browser support, the e-mail address can be automatically validated when submitted. Some
smart phones recognize the email type, and adds ".com" to the keyboard to match email
input.</p>
        E-mail: <input type="email" name="email">
        <p>Defines a numeric input field. Set restrictions on what numbers are accepted.</p>
        Quantity (between 1 and 5):
        <input type="number" name="quantity" min="1" max="5">
        <p>The input type "range" can be displayed as a slider control. Set restrictions on
what numbers are accepted with the min, max, and step attributes. Default 0 to 100</p>
        Points:
        <input type="range" name="points" min="0" max="10">
        <p>Tel is used for input fields that should contain a telephone number.</p>
        Telephone: <input type="tel" name="usrtel">
        <p>Time allows the user to select a time (no time zone).</p>
        Select a time: <input type="time" name="usr_time">
        <p>URL: is used for input fields that should contain a URL address. Depending on
browser support, the url field can be automatically validated when submitted. Some smart
phones recognize the url type, and adds ".com" to the keyboard to match url input.</p>
        Add your homepage: <input type="url" name="homepage">
        </br></br>
Browsers List: <input list="browsers" name="browser">
                <datalist id="browsers">
                        <option value="Internet Explorer">
                        <option value="Firefox">
                        <option value="Chrome">
                        <option value="Opera">
                        <option value="Safari">
                </datalist>
        </input>
<p>Try selecting more than one file when browsing for files.</p>
Select Files: <input type="file" name="fileChooser" multiple>
        </br></br><input type="submit"/>
</form>
```

```
<p><b>Note:</b>
<ul><li><strong>type="color"</strong> is not supported in Internet Explorer 11 and earlier
versions or Safari 9.1 and earlier versions.</li>
<li><strong>type="date"</strong> is not supported in Internet Explorer 11 and earlier
versions.</li>
<li><strong>type="datetime-local"</strong> is not supported in Firefox, or Internet Explorer
12 and earlier versions.</li>
<li><strong>type="month"</strong> is not supported in Firefox, or Internet Explorer 11 and
earlier versions.</li>
<li><strong>type="week"</strong> is not supported in Firefox, or Internet Explorer 11 and
earlier versions.</li>
<li><strong>type="email"</strong> is not supported in IE9 and earlier.</li>
<li><strong>type="numeric"</strong> is not supported in IE9 and earlier.</li>
<li><strong>type="range"</strong> is not supported in IE9 and earlier.</li>
<li><strong>type="tel"</strong> is only supported in Safari 8 and newer versions.</li>
<li><strong>type="time"</strong> is not supported in IE12 and earlier.</li>
<li><strong>type="url"</strong> is not supported in IE9 and earlier.</li>
<li>The <strong>datalist</strong> tag is not supported in Internet Explorer 9 and earlier
versions, or in Safari</li>
<li><strong>type="file"</strong> The multiple attribute of the input tag is not supported in
Internet Explorer 9 and earlier versions</li>
</ul></p>
</body>
</html>
```

Input Restrictions

Attribute	Description
disabled	Specifies that an input field should be disabled
max**	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min**	Specifies the minimum value for an input field
pattern**	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required**	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step**	Specifies the legal number intervals for an input field
Value	Specifies the default value for an input field

**** - Supported only in HTML5**

Input Attributes

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min
- multiple
- pattern (regexp)
- placeholder
- required
- step
- value
- readonly
- disabled
- maxlength
- size

- max

Form Attributes

• autocomplete	whether a form or input field should have autocomplete on or off.
• novalidate	form data should not be validated when submitted.

Example for Attributes

```
<!DOCTYPE html>
<html>
<body>
<form action="/action.php" id="demoForm" autocomplete="on" method="get" novalidate>
        Value and Read Only Input: <input type="text" name="valuereadonly" value="CSE"
readonly/><br><br>
        Size and Max Length: <input type="text" name="sizelength" size="5"
maxlength="10"/><br><br>
        Disabled Text Box: <input type="text" name="disabletextbox" value="CSE"
disabled/><br><br>
        AutoComplete Off: <input type="email" name="autocomplete"
autocomplete="off"/><br><br>
        Pattern and PlaceHolder: <input type="text" name="country_code" pattern="[A-Za-
z]{2}" title="Three letter country code" placeholder="eg. in, au, nz"/><br><br>
        Set Auto Focus:<input type="text" name="autofocus" autofocus/><br><br>
        <input type="submit" value="User Login" fromnovalidate/><br><br>
        <input type="submit" formmethod="post" formenctype="multipart/form-data"
formaction="/admin.php" value="Admin Login"/><br><br>
        <input type="submit" formtarget="_blank" value="Submit to a new window"/><br><br>
</form>
</body>
</html>
```

SVG (Scalable Vector Graphics)

- W3C Recommendation
- Define Graphics for the Web
- The HTML <svg> element is a container for SVG graphics.
- SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

Example

```
<!DOCTYPE html>
<html>
<body>
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40"
  stroke="green" stroke-width="4" fill="yellow" />
</svg>
<svg width="400" height="100">
  <rect width="400" height="100"
  style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
</svg>
 <svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:15;opacity:0.5" />
</svg>
<svg width="300" height="200">
```

```
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana" x="50" y="86">CSE</text>
  Sorry, your browser does not support inline SVG.
</svg>
</body>
</html>
```

Canvas

- Used to draw graphics on a Web page.
- The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.
- The <canvas> element is only a container for graphics. Use JavaScript to actually draw the graphics.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.
  A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.
- The markup looks like this:
  <canvas id="myCanvas" width="200" height="100"></canvas>

Example

```
<!DOCTYPE html>
<html>
<body>
<canvas id="lineCanvas" width="200" height="100" style="border:1px solid
#000000;"></canvas>
<canvas id="circleCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
<canvas id="textCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
<canvas id="lgradientCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
<canvas id="cgradientCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
<canvas id="imageCanvas1" width="450" height="400" style="border:1px solid #000000;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<p>Image to use:</p>
<img id="sistlogo" src="sist_logo1.jpg" alt="SIST Logo" width="450" height="400">
<p>Canvas to fill:</p>
<canvas id="imageCanvas2" width="450" height="400"
style="border:1px solid #d3d3d3;"></canvas>
<p><button onclick="loadImageCanvas()">Load Image into Canvas</button></p>
```

```
<script>
        var c = document.getElementById("lineCanvas");
        var ctx = c.getContext("2d");
        ctx.moveTo(0,0);
        ctx.lineTo(200,100);
        ctx.stroke();

        var c = document.getElementById("circleCanvas");
        var ctx = c.getContext("2d");
        ctx.beginPath();
        ctx.arc(95,50,40,0,2*Math.PI);
        ctx.stroke();

        var c = document.getElementById("textCanvas");
        var ctx = c.getContext("2d");
        ctx.font = "30px Arial";
        ctx.fillText("SIST",10,35);
        ctx.strokeText("CSE",10,75);

        var c = document.getElementById("lgradientCanvas");
        var ctx = c.getContext("2d");
        var grd = ctx.createLinearGradient(0,0,200,0);
        grd.addColorStop(0,"red");
        grd.addColorStop(1,"white");
        ctx.fillStyle = grd;
        ctx.fillRect(10,10,150,80);

        var c = document.getElementById("cgradientCanvas");
        var ctx = c.getContext("2d");
        var grd = ctx.createRadialGradient(75,50,5,90,60,100);
        grd.addColorStop(0,"red");
        grd.addColorStop(1,"white");
        ctx.fillStyle = grd;
        ctx.fillRect(10,10,150,80);

        var c = document.getElementById("imageCanvas1");
        var ctx = c.getContext("2d");
        var img1=new Image();
        img1.onload = function() {
                ctx.drawImage(img1, 0, 0);
        };
        img1.src="sist_logo1.jpg";

        function loadImageCanvas() {
                var c = document.getElementById("imageCanvas2");
                var ctx = c.getContext("2d");
                var img = document.getElementById("sistlogo");
                ctx.drawImage(img,10,10);
        }
</script>
</body>
</html>
```

Difference between SVG and Canvas

SVG	Canvas
Language for describing 2D graphics in XML. Save the resulting image as .png or .jpg	Draws 2D graphics, on the fly
Resolution dependent	Resolution independent
No support for event handlers	Support for event handlers
Poor text rendering capabilities	Best suited for applications with large rendering areas (Google Maps) Slow rendering if complex
Well suited for graphic-intensive games	Not suited for game applications

Google Maps

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Google Map</h1>
<div id="map" style="width:400px;height:400px">
<script>
function myMap() {
        // Set Map Properties
   var mapOptions = {
        // Where to center the Map using the latitude and longitude
        center: new google.maps.LatLng(12.87, 80.21),
        // Zoom to the location
        zoom: 15,
        // ROADMAP, SATELLITE, HYBRID, TERRAIN
        mapTypeId: google.maps.MapTypeId.HYBRID
   }
        var map = new google.maps.Map(document.getElementById("map"), mapOptions);
}
</script>
<!-- To get a new API key for Google Map use the link:
https://developers.google.com/maps/documentation/javascript/get-api-key
-->
<script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDjfMhzcBFlvsD15GR_Ongye4Rp
_xFDYlY&callback=myMap"></script>
</body>
</html>
```

HTML Multimedia

- Multimedia on the web is sound, music, videos, movies, and animations.
- Web pages often contain multimedia elements of different types and formats.
- Examples: Images, music, sound, videos, records, films, animations, and more.
- Multimedia elements (like audio or video) are stored in media files.
- The most common way to discover the type of a file, is to look at the file extension.
- Multimedia files have formats and different extensions like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

Video Formats

Format	File	Description
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Used to be supported by all browsers, but it is not supported in HTML5
AVI	.avi	AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
WMV	.wmv	WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
QuickTime	.mov	QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers.
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. It is still used for online video and Internet TV, but does not play in web browsers.
Flash	.swf .flv	Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers.
Ogg	.ogg	Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5.
WebM	.webm	WebM. Developed by the web giants, Mozilla, Opera, Adobe, and Google. Supported by HTML5.
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Based on QuickTime. Commonly used in newer video cameras and TV hardware. Supported by all HTML5 browsers. Recommended by YouTube.

Audio Formats

Format	File	Description
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface). Main format for all electronic music devices like synthesizers and PC sound cards. MIDI files do not contain sound, but digital notes that can be played by electronics. Plays well on all computers and music hardware, but not in web browsers.
RealAudio	.rm .ram	RealAudio. Developed by Real Media to allow streaming of audio with low bandwidths. Does not play in web browsers.
WMA	.wma	WMA (Windows Media Audio). Developed by Microsoft. Commonly used in music players. Plays well on Windows computers, but not in web browsers.
AAC	.aac	AAC (Advanced Audio Coding). Developed by Apple as the default format for iTunes. Plays well on Apple computers, but not in web browsers.
WAV	.wav	WAV. Developed by IBM and Microsoft. Plays well on Windows, Macintosh, and Linux operating systems. Supported by HTML5.
Ogg	.ogg	Ogg. Developed by the Xiph.Org Foundation. Supported by HTML5.
MP3	.mp3	MP3 files are actually the sound part of MPEG files. MP3 is the most popular format for music players. Combines good compression (small files) with high quality. Supported by all browsers.
MP4	.mp4	MP4 is a video format, but can also be used for audio. MP4 video is the upcoming video format on the internet. This leads to automatic support for MP4 audio by all browsers.

Video
Before HTML5, a video could only be played in a browser with a plug-in (like flash).
The HTML5 <video> element specifies a standard way to embed a video in a web page.

```
<!DOCTYPE html>
<html>
<body>
<video width="320" height="240" controls autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Your browser does not support the video tag.
</video>
</body>
</html>
```

- The controls attribute adds video controls, like play, pause, and volume.
- If height and width are not set, the page might flicker while the video loads.
- The <source> element allows to specify an alternative video files which the browser may choose from. The browser will use the first recognized format.
- The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.
- To start a video automatically use the autoplay attribute.

User defined Video controls

```
<!DOCTYPE html>
<html>
<body>
<div style="text-align:center">
  <button onclick="playPause()">Play/Pause</button>
  <button onclick="makeBig()">Big</button>
  <button onclick="makeSmall()">Small</button>
  <button onclick="makeNormal()">Normal</button>
  <br><br>
  <video id="video1" width="420">
    <source src="mov_bbb.mp4" type="video/mp4">
    <source src="mov_bbb.ogg" type="video/ogg">
    Your browser does not support HTML5 video.
  </video>
</div>
<script>
var myVideo = document.getElementById("video1");
function playPause() {
    if (myVideo.paused)
        myVideo.play();
    else
        myVideo.pause();
}
function makeBig() {
    myVideo.width = 560;
}
function makeSmall() {
    myVideo.width = 320;
}
```

```
function makeNormal() {
    myVideo.width = 420;
}
</script>
</body>
</html>
```

Audio
To play an audio file in HTML, use the <audio> element

```
<!DOCTYPE html>
<html>
<body>
<audio controls>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
</body>
</html>
```

Plugins

- The <object> element is supported by all browsers.
- The <object> element defines an embedded object within an HTML document.
- It is used to embed plug-ins (like Java applets, PDF readers, Flash Players) in Web pages.
- The <embed> element is supported in all major browsers.
- The <embed> element also defines an embedded object within an HTML document.
- Web browsers have supported the <embed> element for a long time. However, it has not been a part of the HTML specification before HTML5.

```
<!DOCTYPE html>
<html>
<body>
Flash File:
        <object width="400" height="50" data="bookmark.swf"></object>
        <embed width="400" height="50" src="bookmark.swf">
HTML Page:
        <object width="100%" height="500px" data="snippet.html"></object>
        <embed width="100%" height="500px" src="snippet.html">
Load Image :
        <object data="sist_logo.jpg"></object>
        <embed src=" sist_logo.jpg ">
</body>
</html>
```

Youtube

Autoplay
Value 0 (default): The video will not play automatically when the player loads.
Value 1: The video will play automatically when the player loads.

PlayList
A comma separated list of videos to play (in addition to the original URL).

Loop
Value 0 (default): The video will play only once.
Value 1: The video will loop (forever).

Controls
Value 0: Player controls does not display.
Value 1 (default): Player controls display.

```
<object width="420" height="315"
data="https://www.youtube.com/embed/jyDNdeqNTb0"></object>
<embed width="420" height="315" src="https://www.youtube.com/embed/jyDNdeqNTb0">
<iframe width="420" height="345" src="https://www.youtube.com/embed/jyDNdeqNTb0
?autoplay=1"></iframe>
```

GeoLocation API

- The HTML Geolocation API is used to get the geographical position of a user.
- Since this can compromise privacy, the position is not available unless the user approves it.
- Geolocation is most accurate for devices with GPS

Example

- Check if Geolocation is supported
- If supported, run the getCurrentPosition() method. If not, display a message to the user
- If the getCurrentPosition() method is successful, it returns a coordinates object to the function specified in the parameter (showPosition)
- showPosition() function outputs the Latitude and Longitude
- showError() function outputs the error (if any)
- options – set accuracy, timeout, refresh status can be set

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to get your coordinates.</p>
<button onclick="getLocation()">Try It</button>
<p id="pos1"></p>
<p id="pos2"></p>
<div id="mapholder"></div>
<script>
var x = document.getElementById("pos1");
var x1 = document.getElementById("pos2");
var options = {
  enableHighAccuracy: true,
  timeout: 5000,
  maximumAge: 0
};
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition, showError, options);
    } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
    }
}
```

```
function showError(error) {
   switch(error.code) {
      case error.PERMISSION_DENIED:
         x.innerHTML = "User denied the request for Geolocation."
         break;
      case error.POSITION_UNAVAILABLE:
         x.innerHTML = "Location information is unavailable."
         break;
      case error.TIMEOUT:
         x.innerHTML = "The request to get user location timed out."
         break;
      case error.UNKNOWN_ERROR:
         x.innerHTML = "An unknown error occurred."
         break;
   }
}
/* API Key - https://developers.google.com/maps/documentation/static-maps/ */
function showPosition(position) {
   var latlon = position.coords.latitude + "," + position.coords.longitude;
   var img_url = "https://maps.googleapis.com/maps/api/staticmap?center="

+latlon+"&zoom=14&size=400x300&key=AIzaSyDR4GC3N3pmR6nOVMuDMsqYDgb1V9BL
de0";
   document.getElementById("mapholder").innerHTML = "<img src='"+img_url+"'>";
         x.innerHTML = "Latitude: " + position.coords.latitude +
   "<br>Longitude: " + position.coords.longitude;
         x1.innerHTML = "Latitude: " + position.coords.latitude +
   "<br>Longitude: " + position.coords.longitude;
}
function getLocation() {
   if (navigator.geolocation) {
      navigator.geolocation.watchPosition(showPosition);
   } else {
      x1.innerHTML = "Geolocation is not supported by this browser.";
   }
}
</script>
</body>
</html>
```

Other Functions

- watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
- clearWatch() - Stops the watchPosition() method.
- Requires an accurate GPS device to test this.

Mark Map

```
<script
src="https://maps.google.com/maps/api/js?key=AIzaSyCP3McXvXtA0kPEGM6OOjstZvOWh
cbIJ5s"></script>
var lat = position.coords.latitude;
var lon = position.coords.longitude;
var latlon = new google.maps.LatLng(lat, lon);
var myOptions = {
   center:latlon,zoom:14,
   mapTypeId:google.maps.MapTypeId.ROADMAP,
   mapTypeControl:false,
   navigationControlOptions:{style:google.maps.NavigationControlStyle.SMALL}
   }
   var mapholder = document.getElementById('mapholder');
   mapholder.style.height = '250px';
   mapholder.style.width = '500px';
   var map = new google.maps.Map(document.getElementById("mapholder"), myOptions);
   var marker = new google.maps.Marker({position:latlon,map:map,title:"You are here!"});
```

Drag and Drop
Common feature.
"grab" an object and drag it to a different location.
Any element is draggable

Example:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#div1 {
   width: 400px;
   height: 400px;
   float: left;
   padding: 10px;
   border: 1px solid #aaaaaa;
}
</style>
<script>
function allowDrop(ev) {
   ev.preventDefault();
}
function drag(ev) {
   ev.dataTransfer.setData("logo", ev.target.id);
}
function drop(ev) {
   ev.preventDefault();
   var data = ev.dataTransfer.getData("logo");
   ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>
<p>Drag the SIST LOGO into the rectangle:</p>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<img id="drag1" src="sist_logo1.jpg" draggable="true" ondragstart="drag(event)" width="400"
height="400">
</body>
</html>
```

- To make an element draggable, set the draggable attribute to true:
- <img draggable="true">
- The ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.
- The dataTransfer.setData() method sets the data type and the value of the dragged data
- The ondragover event specifies where the dragged data can be dropped.
- By default, data/elements cannot be dropped in other elements. To allow a drop, have to prevent the default handling of the element.
- This is done by calling the event.preventDefault() method for the ondragover event.

Web Storage

- With web storage, web applications can store data locally within the user's browser.
- Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
- Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
- Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.
- HTML web storage provides two objects for storing data on the client:
    - window.localStorage - stores data with no expiration date
    - window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)

Before using web storage, check browser support for localStorage and sessionStorage:

```
if (typeof(Storage) !== "undefined") {
        // Code for localStorage/sessionStorage.
} else {
   // Sorry! No Web Storage support..
}
```

Local Storage
Example:

```
<!DOCTYPE html>
<html>
<body>
<div id="result"></div>
<script>
// Check browser support
if (typeof(Storage) !== "undefined") {
   // Store
   localStorage.setItem("deptname", "CSE");
        /*
                //Other method:
                // Store
                localStorage. deptname = "CSE";
                // Retrieve
                document.getElementById("result").innerHTML = localStorage. deptname;
        */
   // Retrieve
   document.getElementById("result").innerHTML = localStorage.getItem("deptname");
} else {
   document.getElementById("result").innerHTML = "Sorry, your browser does not support
Web Storage...";
}
</script>
</body>
</html>
```

- Create a localStorage name/value pair with name="deptname" and value="CSE"
- Retrieve the value of "deptname" and insert it into the element with id="result"
- The syntax for removing the "deptname" localStorage item is as follows:
    - localStorage.removeItem("deptname");
- Name/value pairs are always stored as strings. Have to convert them to another format when required.

Example (To count the number of clicks made in a button):

```
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
   if(typeof(Storage) !== "undefined") {
      if (localStorage.clickcount) {
         localStorage.clickcount = Number(localStorage.clickcount)+1;
      } else {
         localStorage.clickcount = 1;
      }
      document.getElementById("result").innerHTML = "You have clicked the button " +
localStorage.clickcount + " time(s).";
   } else {
      document.getElementById("result").innerHTML = "Sorry, your browser does not support
web storage...";
   }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter will continue to count
(is not reset).</p>
</body>
</html>
```

SessionStorage Object

The sessionStorage object is equal to the localStorage object, except that it stores the data for only one session. The data is deleted when the user closes the specific browser tab.

The following example counts the number of times a user has clicked a button, in the current session:

Example:

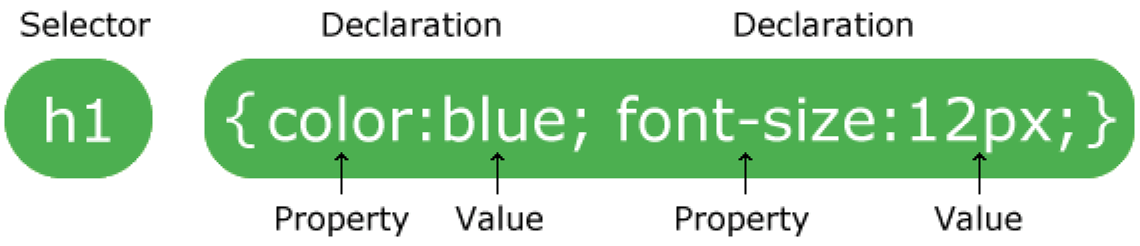
```
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
   if(typeof(Storage) !== "undefined") {
      if (sessionStorage.clickcount) {
         sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
      } else {
         sessionStorage.clickcount = 1;
      }
      document.getElementById("result").innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session.";
   } else {
      document.getElementById("result").innerHTML = "Sorry, your browser does not support
web storage...";
   }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter is reset.</p>
</body>
</html>
```

Cascading Style Sheets (CSS)

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files
- CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.
- HTML was NEVER intended to contain tags for formatting a web page!
- HTML was created to **describe the content** of a web page, like:
    - <h1>This is a heading</h1>
    - <p>This is a paragraph.</p>
- When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.
- To solve this problem, the World Wide Web Consortium (W3C) created CSS.
- CSS removed the style formatting from the HTML page!
- The style definitions are normally saved in external .css files. With an external stylesheet file, you can change the look of an entire website by changing just one file!

Syntax

A CSS rule-set consists of a selector and a declaration block:



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

Example:

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

```
<p>Department of Computer Science and Engineering.</p>
</body>
</html>
```

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more

Element Selector
The element selector selects elements based on the element name.
Can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color – refer previous example)

id Selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element should be unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.
- The style rule below will be applied to the HTML element with id="para1":
- An id name cannot start with a number

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
   text-align: center;
   color: blue;
}
</style>
</head>
<body>
<p id="para1">SIST-DU</p>
<p>DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING</p>
</body>
</html>
```

Class Selector
The class selector selects elements with a specific class attribute.
To select elements with a specific class, write a period (.) character, followed by the name of the class.
In the example below, all HTML elements with class="center" will be red and center-aligned:
You can also specify that only specific HTML elements should be affected by a class.
For example, only <p> elements with class="center" will be center-aligned can be set.

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

```
</style>
</head>
<body>
<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>
</body>
</html>
```

HTML elements can also refer to more than one class.
In the example below, the <p> element will be styled according to class="center" and to class="large":

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
   text-align: center;
   color: red;
}
p.large {
   font-size: 300%;
}
</style>
</head>
<body>
<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>
<p class="center large">This paragraph will be red, center-aligned, and in a large font-size.</p>
</body>
</html>
```

Selectors can be grouped:
Example:

```
h1 {
   text-align: center;
   color: red;
}
h2 {
   text-align: center;
   color: red;
}
p {
   text-align: center;
   color: red;
}
```

Can be modified to:

```
h1, h2, p {
   text-align: center;
   color: red;
}
```

Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.
Comments are ignored by browsers.
A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

```
p {
   color: red;
   /* This is a single-line comment */
   text-align: center;
}

/* This is
a multi-line
comment */
```

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.
There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

External Style Sheet

- With an external style sheet, the look of an entire website.
- Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section.
- An external style sheet can be written in any text editor.
- The file should not contain any html tags.
- The style sheet file must be saved with a .css extension.

Example

mystyle.css

```
body {
   background-color: lightblue;
}

h1 {
   color: navy;
   margin-left: 20px;
}
```

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Internal Style Sheet
An internal style sheet may be used if one single page has a unique style.
Internal styles are defined within the <style> element, inside the <head> section of an HTML page

Example
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
   background-color: linen;
}
h1 {
   color: maroon;
   margin-left: 40px;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Inline Style
- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element.
- The style attribute can contain any CSS property.
- An inline style loses many of the advantages of a style sheet (by mixing content with presentation)

Example
```
<!DOCTYPE html>
<html>
<body>
<h1 style="color:blue;margin-left:30px;">Heading Element with CSS Style Applied.</h1>
<p>Paragraph Element without any style.</p>
</body>
</html>.
```

Cascading Order
What style will be used when there is more than one style specified for an HTML element?
Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:
1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style (inside a specific HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or a browser default value.

Example

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
body {background-color: linen;}
</style>
</head>
<body style="background-color: lightcyan">

<h1>Multiple Styles Will Cascade into One</h1>
<p>In this example, the background color is set inline, in an internal stylesheet, and in an external stylesheet.</p>
<p>Try experimenting by removing styles to see how the cascading stylesheets work. (try removing the inline first, then the internal, then the external)</p>

</body>
</html>
```

Colours, Background, Borders

```
<!DOCTYPE html>
<html>
<style>
body {
   /*background-image: url("sist_logo.jpg");
        background-position: right top;
        background-attachment: fixed; *//* Will not scroll with the page */
        /*background-repeat: repeat-x;  repeat - horizontally
        background-repeat: repeat-x;  repeat - vertically
        background-repeat: no-repeat; no repeat*/

        /* or the above styles can be given using one single attribute:
                background: #ffffff url("sist_logo.jpg") no-repeat right top;
        */
}
        p.dotted {
                border-style: dotted;
                border-width: 5px;
                border-color: red;
        }
        p.mixed {
                border-top-style: dotted;
                border-right-style: solid;
                border-bottom-style: dotted;
                border-left-style: solid;
        }
        p.dashed {border-style: dashed;}
        p.solid {
                border-style: solid;
                border-radius: 15px;
                border-bottom: 6px solid red;
        }
        p.double {border-style: double;}
        p.groove {border-style: groove;}
        p.ridge {border-style: ridge;}
        p.inset {border-style: inset;}
        p.outset {border-style: outset;}
        p.none {border-style: none;}
        p.hidden {border-style: hidden;}
        p.mix {border-style: dotted dashed solid double;}
</style>
<body>

Background Colours:
<h1 style="background-color:Tomato;">Tomato</h1>
<h1 style="background-color:Orange;">Orange</h1>
<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>
<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>
<h1 style="background-color:Gray;">Gray</h1>
<h1 style="background-color:SlateBlue;">SlateBlue</h1>
<h1 style="background-color:Violet;">Violet</h1>
<h1 style="background-color:LightGray;">LightGray</h1>

Text Colours:
<h3 style="color:Tomato;">SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY</h3>
<p style="color:DodgerBlue;">SCHOOL OF COMPUTING</p>
<p style="color:MediumSeaGreen;">DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING</p>

Border Colours:
<h1 style="border: 2px solid Tomato;">SoC</h1>
<h1 style="border: 2px solid DodgerBlue;">SIST</h1>
<h1 style="border: 2px solid Violet;">CSE</h1>

Colours using RGB/HEX/HSL:
<p>Same as color name "Tomato":</p>
rgb(red, green, blue)
<h1 style="background-color:rgb(255, 99, 71);">rgb(255, 99, 71)</h1>
<h1 style="background-color:#ff6347;">#ff6347</h1>
hsl(hue, saturation, lightness)
Hue:
Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.
Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white.
Saturation:
Saturation can be describe as the intensity of a color. 100% is pure color, no shades of gray.
50% is 50% gray, but you can still see the color. 0% is completely gray, you can no longer
see the color.
Lightness
The lightness off a color can be described as how much light you want to give the color,
where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means
full lightness (white).

<h1 style="background-color:hsl(9, 100%, 64%);">hsl(9, 100%, 64%)</h1>
<p>Same as color name "Tomato", but 50% transparent:</p>
```

```
rgba(red, green, blue, alpha)
<h1 style="background-color:rgba(255, 99, 71, 0.5);">rgba(255, 99, 71, 0.5)</h1>
hsla(hue, saturation, lightness, alpha)
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">hsla(9, 100%, 64%, 0.5)</h1>
<p>In addition to the predefined color names, colors can be specified using RGB, HEX, HSL,
or even transparent colors using RGBA or HSLA color values.</p>

Borders:
<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>
<p class="mixed">A Mixed border with margins.</p>

</body>
</html>
```

CSS Margins

- The CSS margin properties are used to create space around elements, outside of any defined borders.
- CSS provides full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).
- CSS has properties for specifying the margin for each side of an element:
  - margin-top
  - margin-right
  - margin-bottom
  - margin-left
- All the margin properties can have the following values:
  - auto - the browser calculates the margin. horizontally center the element within its container. The element will then take up the specified width, and the remaining space will be split equally between the left and right margins
  - length - specifies a margin in px, pt, cm, etc.
  - % - specifies a margin in % of the width of the containing element
  - inherit - specifies that the margin should be inherited from the parent element
  - Negative values are allowed.

Padding

- The CSS padding properties are used to generate space around an element's content, inside of any defined borders.
- CSS provides full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).
- CSS has properties for specifying the padding for each side of an element:
  - padding-top
  - padding-right
  - padding-bottom
  - padding-left

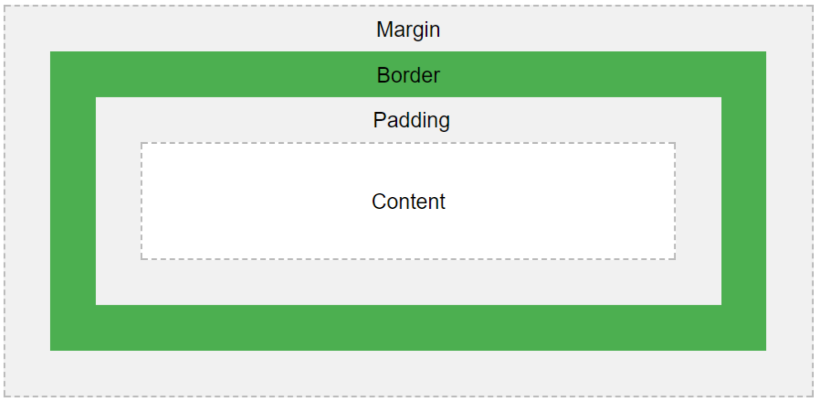
- All the padding properties can have the following values:
    - length - specifies a padding in px, pt, cm, etc.
    - % - specifies a padding in % of the width of the containing element
    - inherit - specifies that the padding should be inherited from the parent element
    - Negative values are not allowed.

Setting height and width
The height and width properties are used to set the height and width of an element.
The height and width can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in length values, like px, cm, etc., or in percent (%) of the containing block.
The max-width property is used to set the maximum width of an element.
The max-width can be specified in length values, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).
The problem with the <div> above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.
Using max-width instead, in this situation, will improve the browser's handling of small windows.
Tip: Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

Example
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   border: 1px solid black;
   margin-top: 100px;
   margin-bottom: 100px;
   margin-right: 150px;
   margin-left: 80px;
   background-color: green;
        color: white;
        font-family: verdana;
        padding-top: 50px;
   padding-right: 30px;
   padding-bottom: 50px;
   padding-left: 80px;
}
p {
        margin: 25px 50px 75px 100px;
        padding: 25px 50px 75px 100px;
}
h1 {
   margin: 25px 50px 75px; /* top, (left and right), bottom */
        padding: 25px 50px 75px;
}
h2 {
        margin: 25px 50px; /* top and bottom, left and right */
        padding: 25px 50px;
}
h4 {
        margin: 25px; /* all four margins */
        padding: 25px;
}
p.mpClass {
        width: 300px;
   padding: 25px;
   box-sizing: border-box;
}
div.hwClass {
   max-width: 500px;
   height: 100px;
   background-color: powderblue;
}
</style>
</head>
<body>
<h1>Margin and Padding - top, (left and right), bottom</h1>
<h2>Margin and Padding - top and bottom, left and right</h2>
<h4>All Four Margins and Padding</h4>
<div>This div element has a Top margin of 100px. <br><br> Right margin of 150px <br><br>
Bottom margin of 100px <br><br> Left margin of 80px. <br><br>
Padding set to: Top: 50px, Right: 30px, Bottom: 50px, Left: 80px.</div>
<p>Shorthand margin</p>
<div class="hwClass"></div>
<p class="mpClass">Test</p>
</body>
</html>
```

The CSS Box Model

- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.
    - Content - The content of the box, where text and images appear
    - Padding - Clears an area around the content. The padding is transparent
    - Border - A border that goes around the padding and content
    - Margin - Clears an area outside the border. The margin is transparent



Example

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

```
div {
   background-color: lightgrey;
   width: 300px;
   border: 25px solid green;
   padding: 25px;
   margin: 25px;
}
</style>
</head>
<body>
<h2>Demonstrating the Box Model</h2>
<p>The CSS box model is essentially a box that wraps around every HTML element. It
consists of: borders, padding, margins, and the actual content.</p>
<div>This text is the actual content of the box. Added a 25px padding, 25px margin and a
25px green border. </div>
</body>
</html>
```

Font – Text Effects

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   border: 1px solid gray;
   padding: 8px;
}

h1 {
   text-align: center;
   text-transform: uppercase;
   color: #4CAF50;
}

p {
   text-indent: 50px;
   text-align: justify;
   letter-spacing: 3px;
        line-height: 2.0;
}

a {
   text-decoration: underline; /*overline, line-through;*/
   color: #008CBA;
}

p.upper {
        text-transform: uppercase; /* lowercase, capitalize */
        letter-spacing: 5px;
        word-spacing: 10px;
        text-shadow: 1px 2px red;
}

p.dirrtl {
        direction: rtl;
}

p.ffly {
        font-family: "Times New Roman", Times, serif, verdana;
        font-style: oblique; /* italics, normal */
        font-size: 40px;
        font-weight: bold;
        font-variant: small-caps;
}
</style>
</head>
<body>

<div>
<h1>text formatting</h1>
<p>This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties.
The paragraph is indented, aligned, and the space between characters is specified. The underline is removed from this colored
<a target="_blank" href="#">"Try it Yourself"</a> link.</p>
<p class="upper">Displays the text in uppercase</p>
<p class="dirrtl"><bdo dir="rtl">Displays the text in left to right. BDO - Bidirectional Text Overried Element</bdo></p>
<p class="ffly">Demonstrate Font Families</p>
</div>
</body>
</html>
```

Transformations – 2D and 3D
CSS transforms allow you to translate, rotate, scale, and skew elements.
A transformation is an effect that lets an element change shape, size and position.
CSS supports 2D and 3D transformations.

2D transformation methods:
- translate()
- rotate()
- scale()
- skewX()
- skewY()
- matrix()

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   width: 300px;
   height: 100px;
   background-color: yellow;
   border: 1px solid black;
   transform: translate(50px,100px); /* Standard syntax */
}
div#myDiv {
   transform: rotate(20deg);
}

div#myScale {
   transform: scale(2,3); /* Standard syntax */
}

div#myScale1 {
   transform: scale(.5,.5); /* Standard syntax */
}
div#myskew {
   transform: skewX(20deg); /* Standard syntax */
        /* skewX, skewY(deg), skew(20deg, 20deg) */
}

div#tMatrix {
         transform: matrix(1, -0.3, 0, 1, 0, 0);
}
</style>
</head>
<body>

<h1>The translate() Method</h1>
<p>The translate() method moves an element from its current position:</p>

<div>
This div element is moved 50 pixels to the right, and 100 pixels down from its current
position.
</div>

<div>
This a normal div element.
</div>

<div id="myDiv">
This div element is rotated clockwise 20 degrees.
</div>

<div id="myScale">
This div element is rotated clockwise 20 degrees.
</div>
<div id="myScale1"> transform: scale(0.5, 0.5);
</div>

<div id="myskew">The skewX() method skews an element along the X-axis by the given
angle.</div>

<div id="tMatrix">
The matrix() method combines all the 2D transform methods into one.
```

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follow:

```
matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())
</div
</body>
</html>
```

Animation

```
<!DOCTYPE html>
<html>
<head>
<style>
/* Draw Box */
#mainBorder {
  position:relative;
  width:500px;
  height:400px;
  margin:0 auto 10px;
  border:1px #aaa solid;
  padding:10px;
}

/* Draw Red Circle */
#redBall {
  width: 100px;
  height: 100px;
  position: absolute;
  top: 160px;
  left: 210px;
  background-color: red;
  border-radius: 50px;
  transition: all 2s ease-in-out;
}

/* Blue Dots */
#blue1, #blue2, #blue3, #blue4, #blue5, #blue6, #blue7, #blue8 {
        width: 20px;
        height: 20px;
        border-radius: 10px;
        position: absolute;
        top: 40px;
        left: 40px;
        background-color: blue;
        transition: all 2s ease-in-out;
        transform: translate3d(0,0,0);
}

#blue1 {
   transition-delay: 0.1s;
}
```

```
#blue2 {
	transition-delay: 0.2s;
}

#blue3 {
	transition-delay: 0.3s;
}

#blue4 {
	transition-delay: 0.4s;
}

#blue5 {
	transition-delay: 0.5s;
}

#blue6 {
	transition-delay: 0.6s;
}

#blue7 {
	transition-delay: 0.7s;
}

#blue8 {
	transition-delay: 0.8s;
}
/* Top Left Diagonal */
#redBall:hover #blue1, #redBall.hover_effect #blue1 {
	transform: translate(-250px,-200px);
}
/* Top */
#redBall:hover #blue2, #redBall.hover_effect #blue2 {
	transform: translate(0,-200px);
}

/* Top Right Diagonal */
#redBall:hover #blue3, #redBall.hover_effect #blue3 {
	transform: translate(250px,-200px);
}

/* Right */
#redBall:hover #blue4, #redBall.hover_effect #blue4 {
	transform: translate(250px, 0);
}

/* Bottom Right Diagonal */
#redBall:hover #blue5, #redBall.hover_effect #blue5 {
	transform: translate(250px,200px);
}

/* Bottom */
#redBall:hover #blue6, #redBall.hover_effect #blue6 {
	transform: translate(0,200px);
}

/* Bottom Left Diagonal */
#redBall:hover #blue7, #redBall.hover_effect #blue7 {
	transform: translate(-250px,200px);
}

/* Left */
#redBall:hover #blue8, #redBall.hover_effect #blue8 {
	transform: translate(-250px,0);
}
</style>
</head>
<body>

<div id="mainBorder">
  <div id="redBall" class="hover">
	<div id="blue1"></div>
	<div id="blue2"></div>
	<div id="blue3"></div>
	<div id="blue4"></div>
	<div id="blue5"></div>
	<div id="blue6"></div>
	<div id="blue7"></div>
	<div id="blue8"></div>
  </div>
</div>
</body>
</html>
```