Chapter 3

❖ HTML 5 Features

1. New Semantic elements added

Semantic Elements: Semantic elements have meaningful names which tells about type of content. For example, header, footer, table, … etc. HTML5 introduces many semantic elements as mentioned below which make the code easier to write and understand for the developer as well as instructs the browser on how to treat them.

- article
- aside
- footer
- header
- nav
- section

2. Audio and Video

Audio and Video tags are the two major addition to HTML5. It allows developers to embed a video or audio on their website. HTML5 video can use CSS and CSS3 to style the video tag. You can change the border, opacity, reflections, gradients, transitions, transformations, and even animations. HTML5 makes adding video super-fast and without having to build a video player.

3. Vector Graphics

This is a new addition to the revised version which has hugely impacted the use of Adobe Flash in websites. Vector graphics are scalable, easy to create and edit. It also supports interactivity and animation. Having a smaller file size makes transferring and loading graphics much faster on the web. That's the reason why many people prefer to use vector graphics. – Canvas, SVG

❖ <!DOCTYPE> declaration

In HTML 5, the declaration is simple:

```
<!DOCTYPE html>
```

In older documents (HTML 4 or XHTML), the declaration is more complicated because the declaration must refer to a DTD (Document Type Definition).

HTML 4.01:

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

A doctype or document type declaration is an instruction that tells the web browser about the markup language in which the current page is written. The Doctype is not an element or tag, it lets the browser know about the version of or standard of HTML or any other markup language that is being used in the document.

List of some common doctype declaration:
• Transitional
• Strict
• Frameset

❖ <aside> tag

- The <aside> tag defines some content aside from the content it is placed in.
- The aside content should be indirectly related to the surrounding content.
    - **Tip:** The <aside> content is often placed as a sidebar in a document.

❖ <article> tag

- The <article> tag specifies independent, self-contained content.
- An article should make sense on its own and it should be possible to distribute it independently from the rest of the site.
- Potential sources for the <article> element:
    - Forum post
    - Blog post
    - News story

Example:

```
<article>
 <p>
  The Disney movie <em>The Little Mermaid</em> was first released to
  theatres in 1989.
 </p>
 <aside>
  <p>The movie earned $87 million during its initial release.</p>
 </aside>
 <p>More info about the movie…</p>
```

```
</article>
```

This example uses <aside> to mark up a paragraph in an article. The paragraph is only indirectly related to the main article content:

We have to add css to make it look proper. You can add it as an inline css using style attribute.

```
<head><style>
aside {
   width: 20%;
   padding: 15px;
   margin-left: 15px;
   float: right;
   font-style: italic;
   background-color: lightgray;
  }
</style></head>
```

<audio> tag

- The <audio> tag is used to embed sound content in a document, such as music or other audio streams.
- The <audio> tag contains one or more <source> tags with different audio sources. The browser will choose the first source it supports.
- The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.
- There are three supported audio formats in HTML: MP3 (MPEG Audio Layer 3), WAV, and OGG.

Ogg is an abbreviation for "Ogging". It is designed to be used with streaming content and can include text, video, music, and metadata.

WAV stands for Waveform Audio File Format. It's a file format for storing audio bitstream on a personal computer. It is a standard digital audio file format used for storing waveform data.

Attribute	Value	Description
autoplay	autoplay	Specifies that the audio will start playing as soon as it is ready
controls	controls	Specifies that audio controls should be displayed (such as a play/pause button etc)
loop	loop	Specifies that the audio will start over again, every time it is finished
muted	muted	Specifies that the audio output should be muted
src	*URL*	Specifies the URL of the audio file

Example

```
<audio controls loop autoplay muted>
  <source src="a1.mp3" type="audio/mp3">
  Your browser does not support the video tag.
</audio>
```

1. **<audio>**:
   - This tag is used to embed audio files on a webpage.
2. **Attributes**:
   - **controls**: This displays the default audio player controls, such as play, pause, volume, and a progress bar, allowing user interaction.
   - **loop**: The audio will loop continuously, replaying automatically when it ends.
   - **autoplay**: The audio will start playing as soon as the page is loaded.
   - **muted**: The audio is muted by default. This is often required for autoplay to function in many browsers.
3. **<source>**:
   - **src="a1.mp3"**: Specifies the file to play, in this case, an MP3 file named a1.mp3.
   - **type="audio/mp3"**: Declares the type of file (MIME type) so the browser knows how to handle it.
4. **Fallback Text**:
   - Your browser does not support the audio tag.: This text is displayed if the user's browser doesn't support the <audio> tag, providing a fallback message.

Without controls attribute audio will not be visible on web page.

<video> tag

- The controls attribute adds video controls, like play, pause, and volume.
- It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.
- The <source> element allows you to specify 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.

There are three different formats that are commonly supported by web browsers – mp4, Ogg, and WebM.

Attributes

- ✓ **Autoplay**: It tells the browser to immediately start downloading the video and play it as soon as it can.
- ✓ **Loop**: It tells the browser to automatically loop the video.
- ✓ **height**: It sets the height of the video in CSS pixels.
- ✓ **width**: It sets the width of the video in CSS pixels.
- ✓ **Controls**: It shows the default video controls like play, pause, volume, etc.
- ✓ **Muted**: It mutes the audio from the video.

The various attributes that can be used with the "source" tag are listed below:

• src: It is used to specify the URL of the video file.

• type: e.g video/mp4, video/ogg

Example

```
<video width="320" height="240" loop autoplay muted controls>
  <source src="v1.mp4" type="video/mp4">
  <source src="v1.ogg" type="video/ogg">
```

```
  Your browser does not support the video tag.
</video>
```

1. **<video>**:
   - This tag is used to embed video files on a webpage.
2. **Attributes**:
   - **width="320" height="240"**: Sets the dimensions of the video player to 320 pixels wide and 240 pixels tall.
   - **loop**: Like the audio tag, this causes the video to loop continuously.
   - **autoplay**: The video starts playing as soon as the page loads.
   - **muted**: The video starts muted by default, which is often necessary for autoplay to work in modern browsers.
   - **controls**: Displays default video controls, such as play, pause, volume, and a fullscreen toggle.
3. **<source>**:
   - There are two <source> tags inside the video element:
     - **src="v1.mp4"**: Specifies the video file in MP4 format.
     - **src="v1.ogg"**: Specifies the same video in OGG format as a fallback in case the browser doesn't support MP4.
   - **type="video/mp4" and type="video/ogg"**: These MIME types declare the formats of the video files.
4. **Fallback Text**:
   - Your browser does not support the video tag.: This text is displayed if the user's browser doesn't support the <video> tag, providing a fallback message.

Add muted after autoplay to let your video start playing automatically (but muted):

<iframe> tag

The <iframe> tag in HTML is used to embed another HTML document within the current document. It allows you to display web pages, multimedia, or other resources like YouTube videos, maps, and forms inside a box on your webpage.

An inline frame is marked up as follows:

```
<iframe src="https://www.w3schools.com" title="W3Schools Free Online Web Tutorials"></iframe>
```

To embed youtube video

src="https://www.youtube.com/embed/Q-qcB_OqKTU"

- ✓ **embed = An embedded video lets you borrow the video from another platform. Visitors can watch the video on your website without leaving the current page.**
- ✓ **video id = Q-qcB_OqKTU**

```
<html>
<head></head>
<body>
<iframe width="500" height="500" src="https://www.youtube.com/embed/Q-qcB_OqKTU" allowfullscreen ></iframe>
</body>
</html>
```

Attribute **allowfullscreen** is set as true if the <iframe> can activate fullscreen mode by calling the requestFullscreen() method

➢ Enabling YouTube autoplay feature:

YouTube's **autoplay** feature can be used to automatically play a video when a user visits that page.

There are two types of parameters that can be used:

- Value 1: The video starts playing automatically when the player loads.
- Value 0 (default case): The video does not play automatically when the player loads.

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

```
</head>
<body>
<iframe width="500" height="500" src="https://www.youtube.com/embed/Q-qcB_OqKTU?autoplay=1"></iframe>
</body>
</html>
```

➢ Enabling / Disabling YouTube controls:

The YouTube Player offers **controls** like play, pause, volume etc. that can be disabled or enabled using the controls parameter. There are two parameters available that can be used:

- Value 1 (default case): Player controls are displayed.
- Value 0: Player controls are not displayed.

```
<html>
<head>
</head>
<body>
<iframe width="500" height="500" src="https://www.youtube.com/embed/Q-qcB_OqKTU?controls=0"></iframe>
</body>
</html>
```

➢ Creating a YouTube playlist to play video in loop and keep video muted:

A playlist of YouTube videos can be created using comma character which separates the list of videos to play.

The **loop** parameter is used to loop the number of playbacks on the videos:

- Value 1: The video will keep on looping again and again.
- Value 0 (default case): The video plays only once.

The **mute** parameter is used to mute the sound of the videos:

- Value 1: The video will keep on mute mode.
- Value 0 (default case): The video will be played with sound.

```
<html>
<head>
</head>
<body>
<iframe width="500" height="500" src="https://www.youtube.com/embed/Q-qcB_OqKTU?playlist=Q-qcB_OqKTU&loop=1&autoplay=1&mute=1"></iframe>
</body>
</html>
```

Example

```
<iframe
src="https://www.youtube.com/embed/LXb3EKWsInQ?playlist=2_kAzyaX7SU,oHdecbMrcbI,LXb3EKWsInQ,kVxTrhojpFI&loop=1&autoplay=1&mute=1&controls=0" title="YouTubevideo playlist" allowfullscreen>
</iframe>
```

Components of the URL Parameters

1. **Base URL**: **https://www.youtube.com/embed/LXb3EKWsInQ**
    - This is the base embed URL for YouTube, where LXb3EKWsInQ is the ID of the initial video that will be displayed.
2. **playlist**:
    - playlist=2_kAzyaX7SU,oHdecbMrcbI,LXb3EKWsInQ,kVxTrhojpFI
    - This parameter specifies the video IDs that form the playlist. After the initial video (LXb3EKWsInQ), the playlist will play the following videos in this order:
        1. LXb3EKWsInQ (repeated)
        2. kVxTrhojpFI
        3. 2_kAzyaX7SU
        4. oHdecbMrcbI
    - When looping is enabled, the playlist will repeat after the last video.
3. **loop=1**:
    - This enables looping, meaning that after the last video in the playlist finishes, the player will automatically start over from the beginning of the playlist.
4. **autoplay=1**:
    - This allows the video to start playing automatically when the page loads, without requiring the user to press the play button.
5. **mute=1**:

- The video will be muted by default. This is often done to allow autoplay functionality, as browsers commonly block autoplay with sound for user experience reasons.

6. **controls=0**:
   - This hides the default YouTube player controls (play, pause, volume, etc.), so the user won't see them in the embedded video. The video will play with minimal UI elements, offering a cleaner look.

- **title="YouTubevideoplaylist"**:
  - This provides a title for the iframe, which is useful for accessibility and SEO purposes. The title helps screen readers and search engines understand the content of the embedded video.

- **allowfullscreen**:
  - This attribute allows users to view the video in fullscreen mode by clicking the fullscreen button (if visible in the embedded player).

SVG - Scalable Vector Graphics

- SVG stands for Scalable Vector Graphics
- SVG is used to define vector-based graphics for the Web
- SVG defines the graphics in XML format
- Every element and every attribute in SVG files can be animated

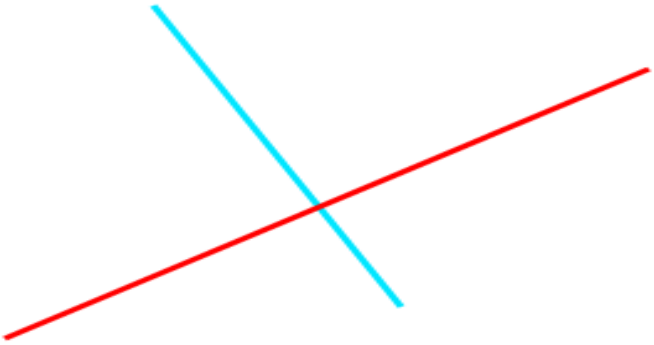
SVG Advantages

Advantages of using SVG over other image formats (like JPEG and GIF) are:

- SVG images can be created and edited with any text editor
- SVG images can be searched, indexed, scripted, and compressed
- SVG images are scalable
- SVG images can be printed with high quality at any resolution
- SVG images are zoomable
- SVG graphics do NOT lose any quality if they are zoomed or resized
- SVG is an open standard
- SVG files are pure XML

1. SVG Line - <line>

```
<svg height="250" width="500" >
  <line x1="100" y1="10" x2="200" y2="200" stroke=”rgb(0,230,255)” stroke-width="3"></line>
  <line x1="300" y1="50" x2="40" y2="220" stroke=”red” stroke-width="3"/>
</svg>
```



Code explanation:

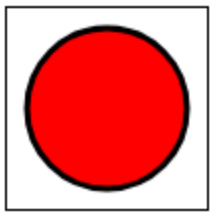
- The x1 attribute defines the start of the line on the x-axis
- The y1 attribute defines the start of the line on the y-axis
- The x2 attribute defines the end of the line on the x-axis
- The y2 attribute defines the end of the line on the y-axis
- The stroke-width attribute defines the width of the border of the line
- The stroke attribute defines the color of the border of the line.

The stroke attribute must be set to a color (e.g., black, red, #000000) for the shape's outline to appear.The **default stroke** is not applied automatically, and an unspecified stroke means no visible stroke at all

Default value is not white. But we need to define a stroke attribute for the shape's outline to appear.

2. SVG Circle

```
<svg height="100" width="100" style="border:1px solid black">
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
</svg>
```



Code explanation:

- The cx and cy attributes define the x and y coordinates of the center of the circle. If cx and cy are omitted, the circle's center is set to (0,0)
- The r attribute defines the radius of the circle.
- The fill attribute defines the fill color of the circle.
- The stroke-width attribute defines the width of the border of the circle.
- The stroke attribute defines the color of the border of the circle.

3. SVG Rectangle - <rect>

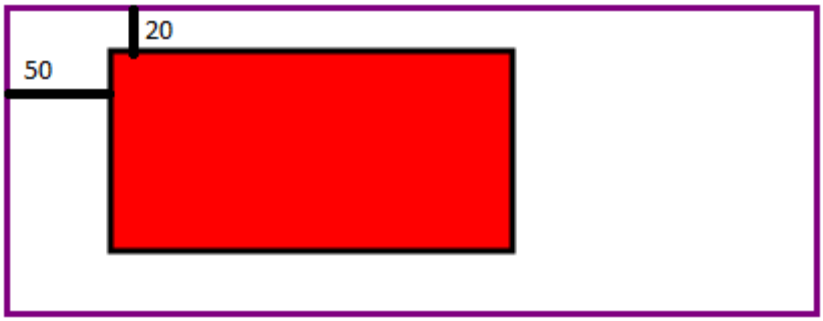
The <rect> element is used to create a rectangle and variations of a rectangle shape:

```
<svg width="400" height="120" style="border:3px solid purple">
  <rect width="300" height="100" stroke="black" stroke-width="3" fill="red"  />
</svg>
```

In above example by default x and y coordinates are (0, 0)

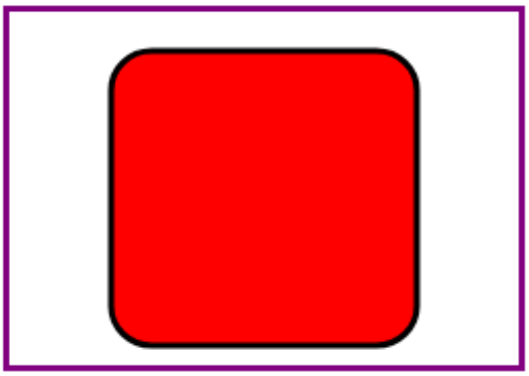
- The width and height attributes of the <rect> element define the height and the width of the rectangle
- The fill attribute defines the fill color of the rectangle.
- The stroke-width attribute defines the width of the border of the rectangle.
- The stroke attribute defines the color of the border of the rectangle.

```
<svg width="400" height="150" style="border:3px solid purple">
 <rect x="50" y="20" width="200" height="100" stroke="black" stroke-width="3" fill="red"  />
</svg>
```



- The x attribute defines the left position of the rectangle (e.g. x="50" places the rectangle 50 px from the left margin)
- The y attribute defines the top position of the rectangle (e.g. y="20" places the rectangle 20 px from the top margin)

```
<svg width="250" height="180" style="border:3px solid purple">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150" stroke="black" stroke-
width="3" fill="red"/>
</svg>
```



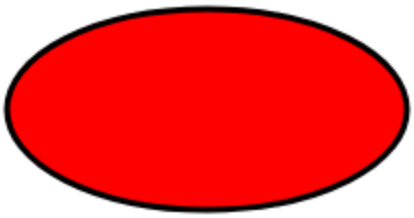
- The rx and the ry attributes rounds the corners of the rectangle

4. SVG Ellipse - <ellipse>

The <ellipse> element is used to create an ellipse.

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

```
<svg height="140" width="500">
  <ellipse cx="200" cy="80" rx="100" ry="50" stroke="black" stroke-width="3" fill="red"/>
</svg>
```



Code explanation:

- The cx attribute defines the x coordinate of the center of the ellipse
- The cy attribute defines the y coordinate of the center of the ellipse
- The rx attribute defines the horizontal radius
- The ry attribute defines the vertical radius.
- The fill attribute defines the fill color of the ellipse.

- The stroke-width attribute defines the width of the border of the ellipse.
- The stroke attribute defines the color of the border of the ellipse.

Logo using ellipse

```
<svg height="140" width="500">
  <ellipse cx="200" cy="80" rx="100" ry="50" stroke="black" stroke-width="3" fill="pink"/>
  <text fill="#000" font-size="50" x="155" y="100" font-family="verdana">LJU</text>
</svg>
```

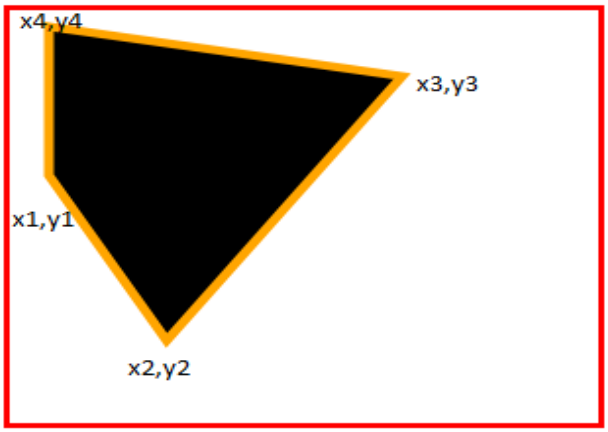
5. SVG Polygon - <polygon>

The <polygon> element is used to create a graphic that contains at least three sides.

Polygons are made of straight lines, and the shape is "closed" (all the lines connect up).

```
<polygon points="x1,y1,x2,y2,x3,y3,x4,y4….xn,yn" stroke="color" stroke-width="5"
fill="black" />
```

```
<svg height="250" width="300" style="border: 3px solid red;">
  <polygon points="20,100,80,200,200,40,20,10" stroke="orange" stroke-width="5"
fill="black" />
</svg>
```

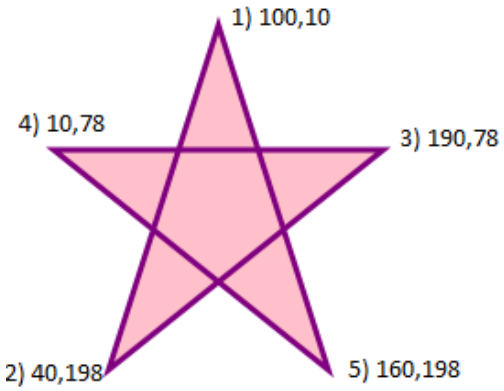


Code explanation:

- The points attribute defines the x and y coordinates for each corner of the polygon
- The fill attribute defines the fill color of the polygon.
- The stroke-width attribute defines the width of the border of the polygon.
- The stroke attribute defines the color of the border of the polygon.

Use the <polygon> element to create a Star:

```
<svg height="210" width="500">
  <polygon points="100,10,40,198,190,78 10,78,160,198"
  stroke="purple" stroke-width="3" fill="pink" fill-rule="nonzero" />
 </svg>
```



✓ The **fill-rule** attribute is a presentation attribute defining the algorithm to use to determine the *inside* part of a shape.

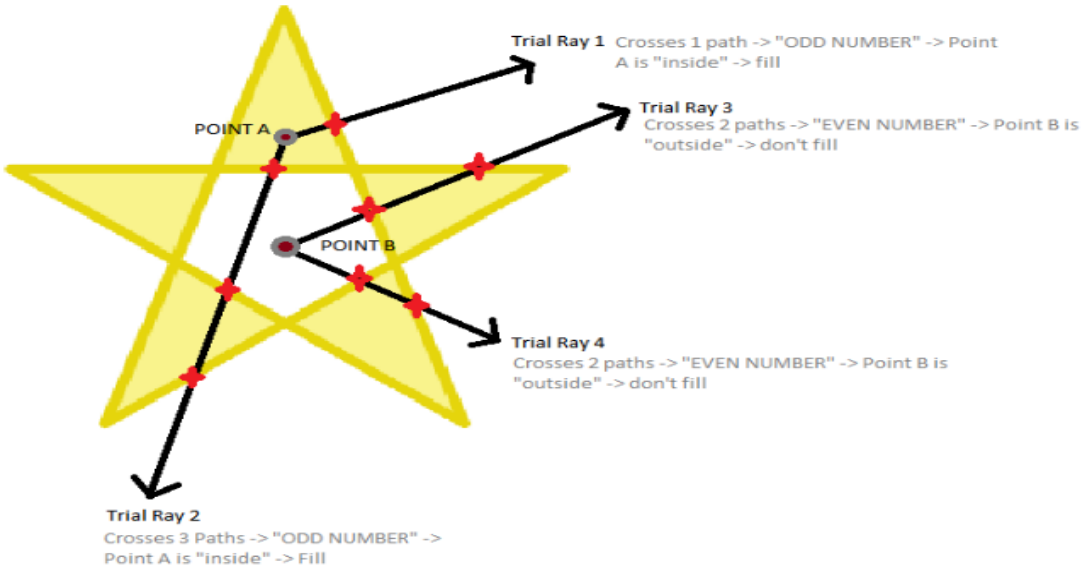
✓ After counting the crossings paths, if the result is zero then the point is outside the path. Otherwise, it is inside**. Default value is "nonzero"**

Change the fill-rule property to "evenodd":

```
<svg height="210" width="500">
  <polygon points="100,10,40,198,190,78 10,78,160,198"
  stroke="purple" stroke-width="3" fill="pink" fill-rule="evenodd" />
 </svg>
```

This is how the mechanism works. Pick a point in the center, draw lines to infinity in any direction - they always cross an even number of path segments -which means that they are "outside" and their areas don't get filled.

Pick a point in the filled triangles - if you draw lines to infinity in any direction - they always cross an odd number of path segments and thus, they are "inside" and their area should be filled.



Note:

- ✓ **If the fill attribute is not specified, then default value is black.**
- ✓ **stroke attribute is necessary to define in line tag, otherwise line will not be visible.**
- ✓ **In other tags like rect, circle, polygon, ellipse etc it will fill the shape with default color black/mentioned color without stroke (border of the shape).**

SVG shapes like <circle>, <rect>, <line> etc do not render a stroke unless a stroke color is explicitly provided. **There is no default stroke automatically applied unless specified**.

Without the stroke attribute, the circle won't render any visible outline because no stroke color is applied. To make the circle visible, you need to explicitly define the stroke color.

Default value is not white. But we need to define a stroke attribute for the shape's outline to appear.

HTML Canvas

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

Canvas Examples

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

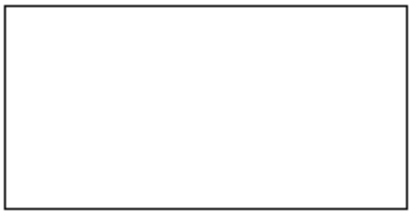
The markup looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

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

Here is an example of a basic, empty canvas:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
```



HTML Image Map

<area> and <map>

- The <area> tag defines an area inside an image map (an image map is an image with clickable areas).
- <area> elements are always nested inside a <map> tag.

Note: The usemap attribute in <img> is associated with the <map> element's name attribute, and creates a relationship between the image and the map.

Attributes:

- **shape**: The shape to be mapped on the image, can be a “rect”, a “circle” or a “poly”.
- **coords**: The coordinates of the shape.
- **href**: The href is the link where the user will be directed to after clicking the mapped portion of the image.
- **alt**: Alternative text for a clickable area in an image map.
- **target**: Context in which to open the link resource.

x1,y1,x2,y2	Specifies the coordinates of the **top-left** and **bottom-right** corner of the rectangle **(shape="rect")**
x,y,radius	Specifies the coordinates of the circle center and the radius **(shape="circle")**
x1,y1,x2,y2,..,xn,yn	Specifies the coordinates of the edges of the polygon. If the first and last coordinate pairs are not the same, the browser will add the last coordinate pair to close the polygon **(shape="poly")**

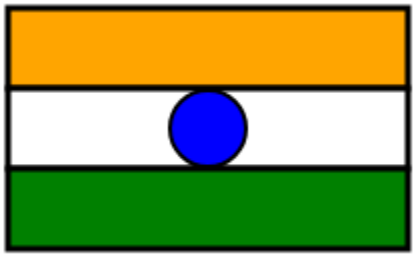
Example:

```
<html>
  <head> <title>Area & Map</title> </head>
  <body>
    <h1> Map & Area </h1>
```

```
 <!-- Image Map Generated by http://www.image-map.net/ -->
<img src="test.jpg" usemap="#image-map">

<map name="image-map">
   <area target="_blank" alt="laptop" title="laptop" href="" coords="0,3,339,523"
shape="rect">

   <area target="_blank" alt="tea" title="tea" href="" coords="437,382,85" shape="circle">

   <area target="_blank" alt="book" title="book" href="" coords=" 465,261,
578,191,668,234,775,393,617,496" shape="poly">

   <area target="_blank" alt="flower" title="flower" href="" coords=" 322,0,324,0
414,149,609,202,783,112,800,3" shape="poly">
</map>
      </body>
</html>
```

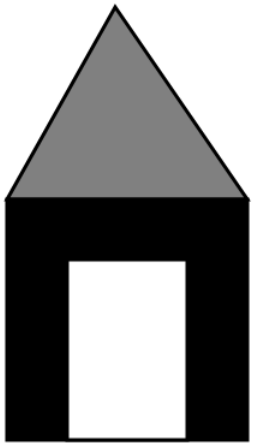
to find coordinates **https://www.image-map.net/** **or** **https://www.imagemap.org**

Example 1



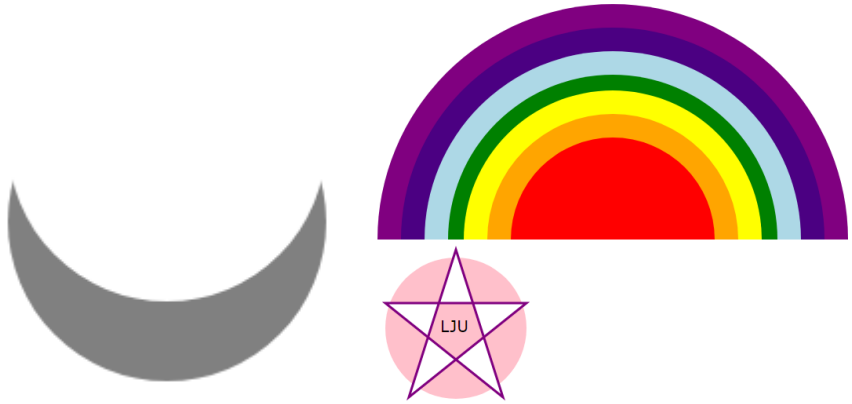
```
<svg height="250" width="500">
    <rect x='20' y='20' width="200" height="40" fill="orange" stroke="#000" stroke-
width="3"/>
    <rect x='20' y='60' width="200" height="40" fill="#fff" stroke="#000" stroke-
width="3"/>
    <rect x='20' y='100' width="200" height="40" fill="green" stroke="#000" stroke-
width="3"/>
    <circle cx="120" cy="80" r="19" stroke="#000" stroke-width="2" fill="blue"/>
  </svg>
```

Example 2



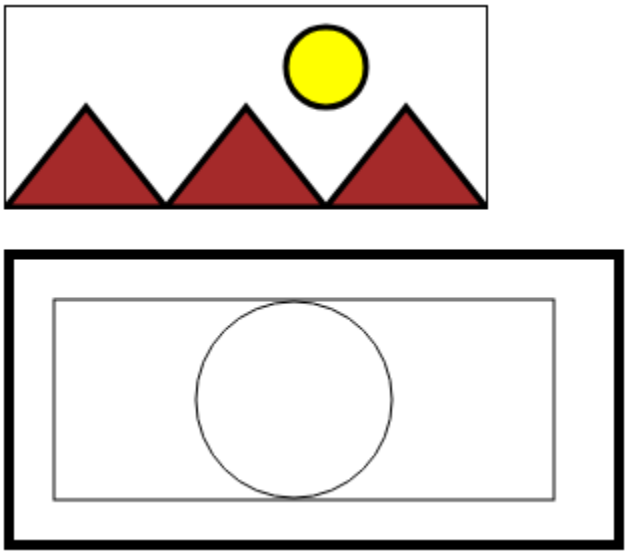
```
 <svg height="450" width="500">
    <polygon points="20,180 110,20 220,180" fill="grey" stroke="black" stroke-width="3" />
    <rect x='20' y='180' width="200" height="200" fill="black" stroke="#000" stroke-
width="3"/>
    <rect x='70' y='230' width="100" height="150" fill="white" stroke="#000" stroke-
width="3"/>
  </svg>
```

Example 3




```
<!—moon -- >
<svg width="200" height="200">
   <circle cx="120" cy="100" r="80" fill="grey"/>
   <circle cx="120" cy="60" r="80" fill="white"/>
</svg>

<!—rainbow half circle -- >
 <svg width="600" height="500">
   <circle cx="300" cy="500" r="300" fill="purple"/>
   <circle cx="300" cy="500" r="270" fill="indigo"/>
   <circle cx="300" cy="500" r="240" fill="lightblue"/>
   <circle cx="300" cy="500" r="210" fill="green"/>
   <circle cx="300" cy="500" r="190" fill="yellow"/>
   <circle cx="300" cy="500" r="160" fill="orange"/>
   <circle cx="300" cy="500" r="130" fill="red"/>
 </svg>

<!—star circle logo -- >
 <svg height="400" width="500">
   <circle cx="100" cy="110" r="90" fill="pink"></circle>
   <polygon points="100,10,40,198,190,78 10,78,160,198"
   stroke="purple" stroke-width="3" fill="white" fill-rule="evenodd" />
   <text fill="#000" font-size="20" x="80" y="115" font-family="verdana">LJU</text>
 </svg>
```

Example 4



```
<svg height="100" width="240" style="border: 1px solid black;">
   <polygon points="0,100,40,50,80,100,80,100,120,50,160,100,160,100,200,50,240,100"
fill="brown" stroke="black" stroke-width="3"/>
   <circle cx="160" cy="30" r="20" fill="yellow" stroke="black" stroke-width="3"/>
</svg>
<br>

<svg height="140" width="300" style="border: 5px solid black;">
   <rect x="20" y="20" width="250" height="100" fill="white" stroke="black" stroke-
width=""/>
   <circle cx="140" cy="70" r="49" fill="#fff" stroke="#000"/>
</svg>
```