Unit-2: HTML5 – Audio and Canvas:

Audio on the Web, Audio Formats, How It Works, All <audio> Attributes, HTML5 Canvas, What is Canvas? Create a Canvas Element, Draw With JavaScript, Understanding Coordinates, More Canvas Examples, HTML5 Web Storage, Storing Data on the Client, The local Storage Object, The session Storage Object, HTML5 Input Types, HTML5 New Input Types, Browser Support, Input Type – email, Input Type – url, Input Type – number, Input Type – range, Input Type - Date Pickers, Input Type – search, Input Type – color.

Audio on the Web

The HTML <audio> element is used to play an audio file on a web page.

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

HTML Audio - How It Works

The controls attribute adds audio controls, like play, pause, and volume.

The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

Audio - Formats

File Format	Media Type
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav

All <audio> Attributes

1. Autoplay : Specifies that the audio will start playing as soon as it is ready

```
<audio controls autoplay>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

2. Controls : Specifies that audio controls should be displayed (such as a play/pause button etc)

Audio controls should include:

- Play
- Pause
- Seeking
- Volume

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

3. Loop: Specifies that the audio will start over again, every time it is finished

```
<audio controls loop>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

4. Muted: Specifies that the audio output should be muted.

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

5. Preload: Specifies if and how the author thinks the audio should be loaded when the page loads.

```
<audio controls preload="none">
 <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

6. Src: Specifies the URL of the audio file

```
<audio src="audio.mp3 " controls>
Your browser does not support the audio element.
</audio>
```

HTML5 Canvas

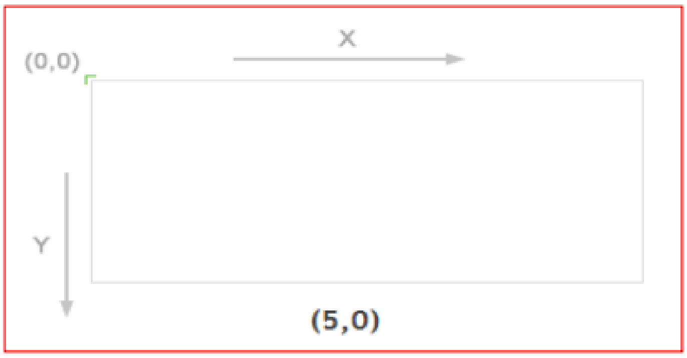
The HTML5 canvas element can be used to draw graphics on the webpage via JavaScript. By default the <canvas> element has 300px of width and 150px of height without any border and content.

Canvas Coordinates

The canvas is a two-dimensional rectangular area.

The coordinates of the top-left corner of the canvas are (0, 0) which is known as origin, and the coordinates of the bottom-right corner are (canvas width, canvas height).

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

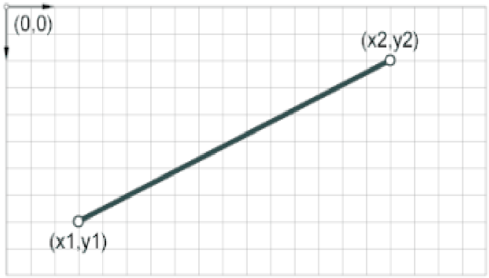


```
<!DOCTYPE html>
<html lang="en"> <head> <meta charset="utf-8"> <title>Drawing on
Canvas</title>
<script>
window.onload = function()
{
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
// draw stuff here
};
</script>
</head>
<body>
<canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

<u>Drawing a Line</u>

To draw a straight line on canva, The essential methods used are moveTo(), lineTo() and the stroke().

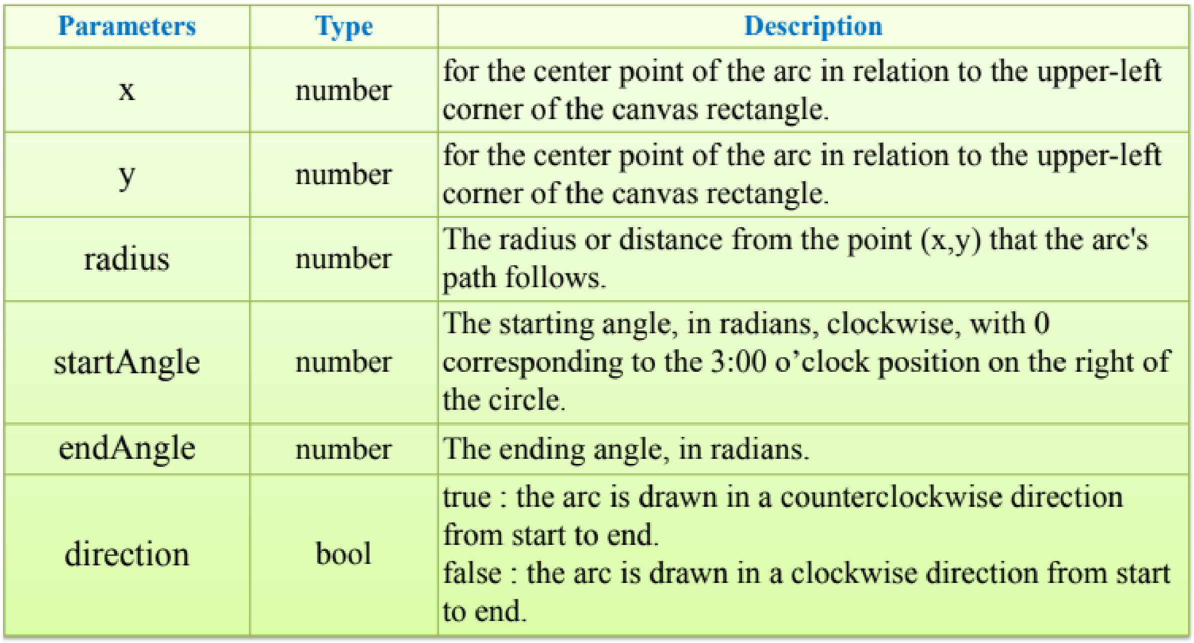
□context.moveTo(0, 0);
// X,Y Coordinators for start point
□context.lineTo(50, 50); //
adds a new point and creates a line TO that point FROM the last specified point in the canvas
□context.stroke();
// draw the path on the canvas



Drawing ARC:

Syntax:

context.arc(centerX, centerY, radius, startingAngle, endingAngle,counterclockwise);



Parameters	Type	Description
x	number	for the center point of the arc in relation to the upper-left corner of the canvas rectangle.
y	number	for the center point of the arc in relation to the upper-left corner of the canvas rectangle.
radius	number	The radius or distance from the point (x,y) that the arc's path follows.
startAngle	number	The starting angle, in radians, clockwise, with 0 corresponding to the 3:00 o'clock position on the right of the circle.
endAngle	number	The ending angle, in radians.
direction	bool	true : the arc is drawn in a counterclockwise direction from start to end. false : the arc is drawn in a clockwise direction from start to end.

var radians = degrees * Math.PI/180

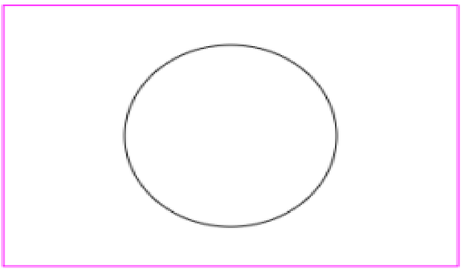
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing an Arc on the Canvas</title>
<style>
canvas { border: 1px solid #F00; }
</style>
<script>
window.onload = function() {
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
context.arc(150, 150, 80, 1.2 * Math.PI, 1.8 * Math.PI, false);
context.stroke();
};
</script>
</head>
<body>
<canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

Drawing a Circle

Syntax: context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);

An arc is a part of a circle. To draw a circle, draw an arc with a starting angle of 0 and ending angle of 2 x Pi.

```
<!DOCTYPE html>
<html lang="en">
<head> <meta charset="utf-8">
<title>Drawing a Circle on the Canvas</title>
<style> canvas { border: 1px solid #F0F; } </style>
<script>
window.onload = function() {
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
context.arc(150, 100, 70, 0, 2 * Math.PI, false);
context.stroke();
};
</script>
</head>
<body>
<canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

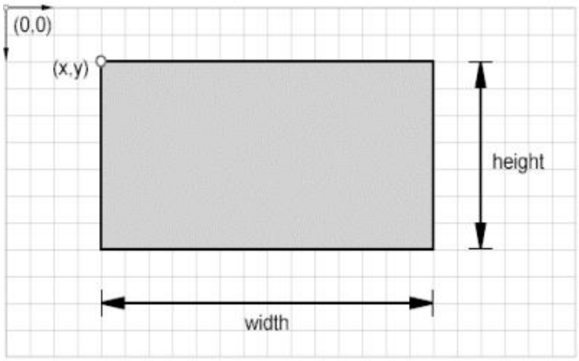


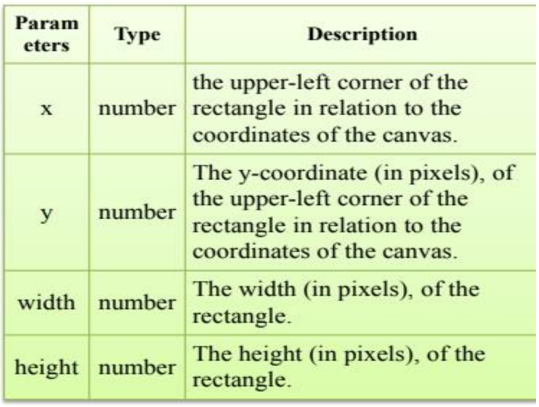
Drawing Rectangle

To draw a rectangle, specify the x and y coordinates (upper-left corner) and the height and width of the rectangle.

There are three rectangle methods:

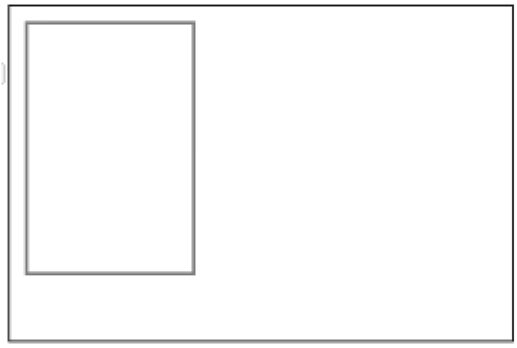
fillRect() strokeRect() clearRect()





Param eters	Type	Description
x	number	the upper-left corner of the rectangle in relation to the coordinates of the canvas.
y	number	The y-coordinate (in pixels), of the upper-left corner of the rectangle in relation to the coordinates of the canvas.
width	number	The width (in pixels), of the rectangle.
height	number	The height (in pixels), of the rectangle.

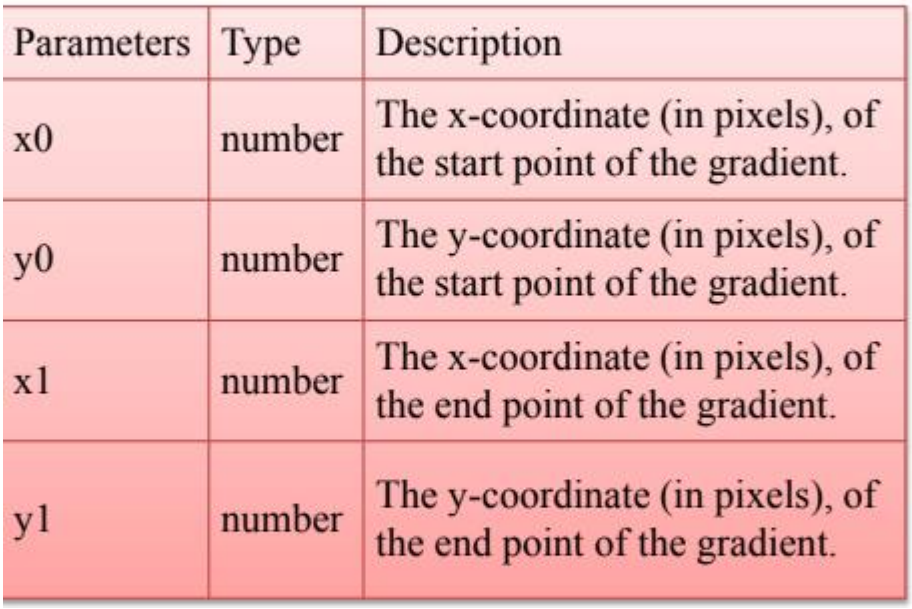
```
<script>
window.onload = function() {
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
context.rect(10, 10, 100, 150);
context.stroke();
};
</script>
<body>
<canvas id="myCanvas" width="300" height="200"></canvas>
</body>
```



Gradients

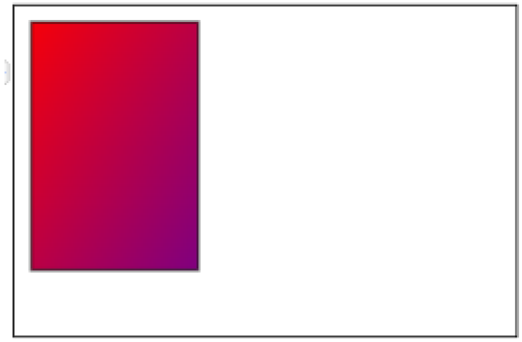
Gradients consist of continuously smooth color transitions along a vector from one color to another, possibly followed by additional transitions along the same vector to other colors. HTML5 canvas (2D) supports two kinds of gradient: linear and radial.

var grd = context.createLinearGradient(startX, startY, endX, endY);



Parameters	Type	Description
x0	number	The x-coordinate (in pixels), of the start point of the gradient.
y0	number	The y-coordinate (in pixels), of the start point of the gradient.
x1	number	The x-coordinate (in pixels), of the end point of the gradient.
y1	number	The y-coordinate (in pixels), of the end point of the gradient.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Filling Linear Gradient inside Canvas Shapes</title>
<style> canvas {border: 1px solid #000; } </style>
<script>
window.onload = function() {
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
context.rect(10, 10, 100, 150);
var grd = context.createLinearGradient(0, 0, canvas.width, canvas.height);
grd.addColorStop(0, '#FF0000');
grd.addColorStop(1, '#0000FF');
context.fillStyle = grd;
context.fill();
context.stroke(); </script>
</head>
<body>
<canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```



HTML5 Web Storage

- The HTML5's web storage feature lets you store some information locally on the user's computer, similar to cookies, but it is faster and much better than cookies.
- The information stored in the web storage isn't sent to the web server as opposed to the cookies where data sent to the server with every request.
- Where cookies let you store a small amount of data (nearly 4KB), the web storage allows you to store up to 5MB of data.

There are two types of web storage, which differ in scope and lifetime:

1. Local storage: The local storage uses the localStorage object to store data for your entire website on a permanent basis.

2. Session storage: The session storage uses the sessionStorage object to store data on a temporary basis, for a single browser window or tab.

The data disappears when session ends i.e. when the user closes that browser window or tab.

localStorage Object

- localStorage.setItem(key, value) :stores the value associated with a key.
- localStorage.getItem(key) :retrieves the value associated with the key.
- localStorage.removeItem(“key") : remove a particular item from the storage if it exists
- localStorage.clear() : if you want to remove the complete storage

```
<script>
if(localStorage) // Check if the localStorage object exists
{
localStorage.setItem("first_name", “ABC"); // Store data
("Hi, " + localStorage.getItem("first_name")); // Retrieve data alert
}
else { alert("Sorry, your browser do not support local storage."); }
</script>
```

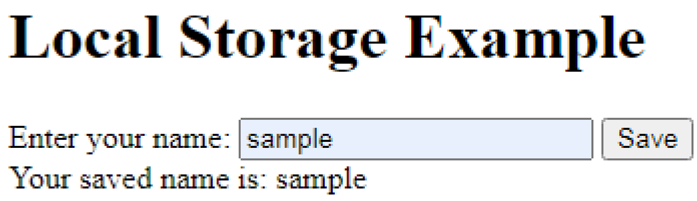
sessionStorage Object

- sessionStorage.setItem(key, value) : stores the value associated with a key.
- sessionStorage.getItem(key) : retrieves the value associated with the key.

```
<script>
// Check if the sessionStorage object exists
if(sessionStorage)
{sessionStorage.setItem("last_name", "Parker"); // Store data
alert("Hi, " + localStorage.getItem("first_name") + " " +
sessionStorage.getItem("last_name")); // Retrieve data
}
else { alert("Sorry, your browser do not support session
storage.");
}
</script>
```

Storing Data on the Client

```
<!DOCTYPE html>
<html>
<head>
  <title>Local Storage Example</title>
</head>
<body>
  <h1>Local Storage Example</h1>

  <!-- Input field for user's name -->
  <label for="name">Enter your name:</label>
  <input type="text" id="name">
  <button onclick="saveName()">Save</button>

  <!-- Display saved name -->
  <div id="savedName"></div>

  <script>
    // Function to save the user's name in localStorage
    function saveName() {
      var name = document.getElementById("name").value;
localStorage.removeItem("user_name");
      localStorage.setItem("user_name", name);
      displaySavedName();
    }

    // Function to retrieve and display the saved name from localStorage
    function displaySavedName() {
      var savedName = localStorage.getItem("user_name");
      var savedNameDisplay = document.getElementById("savedName");

      if (savedName) {
        savedNameDisplay.textContent = "Your saved name is: " + savedName;
      } else {
        savedNameDisplay.textContent = "No name saved.";
      }
    }
    // Display the saved name when the page loads
    displaySavedName();
    localStorage.clear();
  </script>
</body>
</html>
```



Local Storage Example

Enter your name: sample Save

Your saved name is: sample

New Input Types in HTML5

- Color
- date
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Color : The color input type allows the user to select a color from a color picker and returns the color value in hexadecimal format (#rrggbb). If you don't specify a value, the default is #000000, which is black.

```
<input type="color" value="#00ff00" id=“xyz">
```

Date: The date input type allows the user to select a date from a drop-down calendar.

```
<input type="date" value="2019-04-15" id="mydate">
```

Datetime-local :

The datetime-local input type allows the user to select both local date and time, including the year, month, and day as well as the time in hours and minutes.

```
<form>
<label for="mydatetime">Choose Date and Time:</label>
<input type="datetime-local" id="mydatetime">
</form>
```

email :email input type allows the user to enter e-mail address. It is very similar to a standard text input type, but if it is used in combination with the required attribute, the browser may look for the patterns to ensure a properly-formatted e-mail address should be entered.

```
<form>
<label for="myemail">Enter Email Address:</label>
<input type="email" id="myemail" required>
</form>
```

Number :

The number input type can be used for entering a numerical value. You can also restrict the user to enter only acceptable values using the additional attributes min, max, and step.

```
<form>
<label for="mynumber">Enter a Number:</label>
<input type="number" min="1" max="10" step="0.5" id="mynumber">
</form>
```

Range:

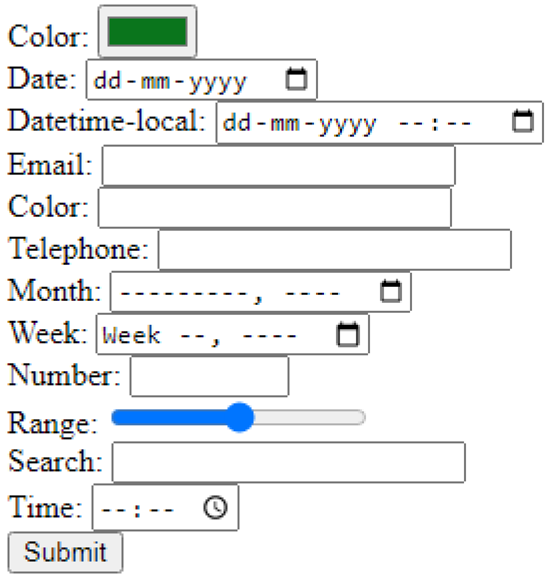
The range input type can be used for entering a numerical value within a specified range. It works very similar to number input, but it offers a simpler control for entering a number. Let's try out the following example to understand how it basically works:

```
<form>
<label for="mynumber">Select a Number:</label>
<input type="range" min="10" max="100" step=“1" id="mynumber">
</form>
```

Search : The search input type can be used for creating search input fields. A search field typically behaves like a regular text field, but in some browsers like Chrome and Safari as soon as you start typing in the search box a small cross appears on the right side of the field that lets you quickly clear the search field.

```
<form>
<label for="mysearch">Search Website:</label>
<input type="search" id="mysearch">
</form>
```

Output:



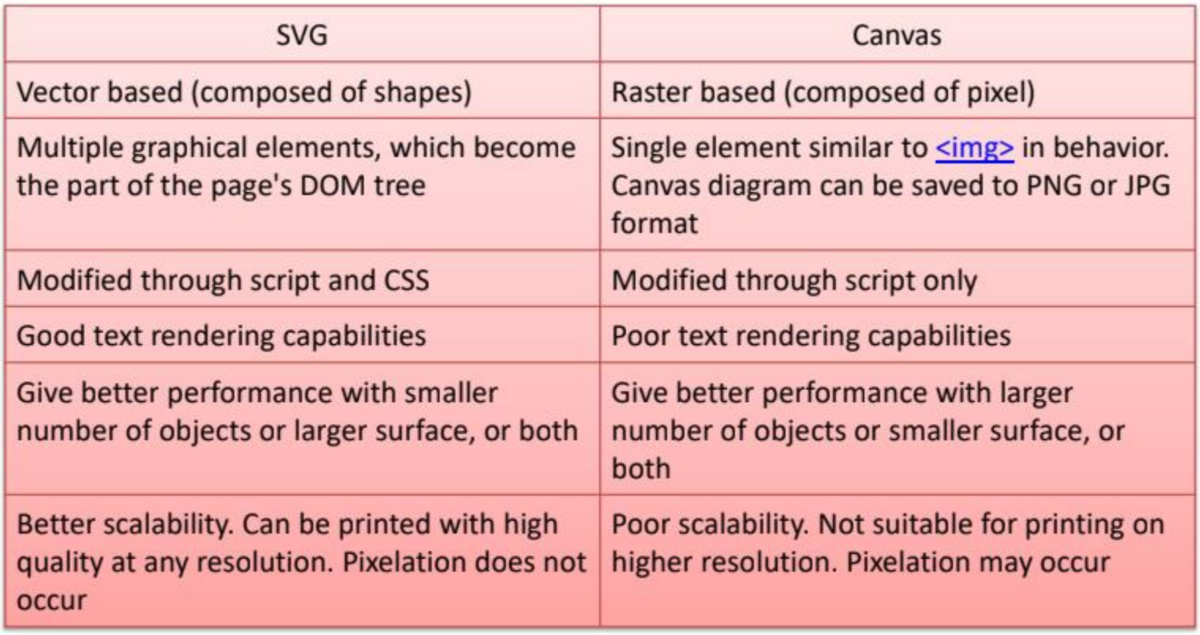
Color: 🟩
Date: dd-mm-yyyy
Datetime-local: dd-mm-yyyy --:--
Email:
Color:
Telephone:
Month: ---------, ----
Week: Week --, ----
Number:
Range:
Search:
Time: --:--

Scalable Vector Graphics (SVG)

The Scalable Vector Graphics (SVG) is an XML-based image format that is used to define two-dimensional vector based graphics for the web. Unlike raster image (e.g. jpg, .gif, .png, etc.), a vector image can be scaled up or down to any extent without losing the image quality.

An SVG image is drawn out using a series of statements that follow the XML schema, means SVG images can be created and edited with any text editor, such as Notepad.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Embedding SVG Into HTML Pages</title>
<style>
svg { border: 1px solid black; }
</style>
</head>
<body>
<svg width="300" height="200">
<text x="10" y="20" style="font-size:14px;">
Your browser support SVG.
</text>
Sorry, your browser does not support SVG.
</svg>
</body>
</html>
```



SVG	Canvas
Vector based (composed of shapes)	Raster based (composed of pixel)
Multiple graphical elements, which become the part of the page's DOM tree	Single element similar to <img> in behavior. Canvas diagram can be saved to PNG or JPG format
Modified through script and CSS	Modified through script only
Good text rendering capabilities	Poor text rendering capabilities
Give better performance with smaller number of objects or larger surface, or both	Give better performance with larger number of objects or smaller surface, or both
Better scalability. Can be printed with high quality at any resolution. Pixelation does not occur	Poor scalability. Not suitable for printing on higher resolution. Pixelation may occur