# HTML5

## HTML Canvas

https://www.w3schools.com/graphics/canvas_intro.asp

The HTML <canvas> element is used to draw graphics on a web page via scripting (usually JavaScript).

The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

In HTML, a <canvas> element looks like this:

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

The <canvas> element must have an id attribute so it can be referred to by JavaScript. The width and height attribute is necessary to define the size of the canvas.

By default, the <canvas> element has no border and no content. To add a border, use a style attribute:

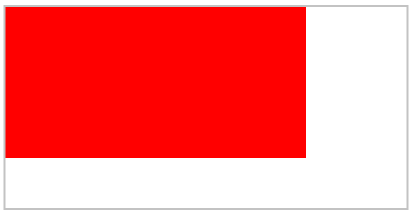
```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid black;"></canvas>
```

### Draw on the Canvas With JavaScript

All drawing on the HTML canvas must be done with JavaScript:

```
<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "red";
ctx.fillRect(0, 0, 150, 75);
</script>
```

**Output:**



**Explanation:**

https://www.w3schools.com/graphics/canvas_drawing.asp

- **Step 1: Find the <canvas> Element:** This is done by using the HTML DOM method getElementById():

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

- **Step 2: Create a Drawing Object:** Secondly, you need a drawing object for the canvas. The getContext() is a built-in HTML object, with properties and methods for drawing:

  ```
  var ctx = canvas.getContext("2d");
  ```

- **Step 3: Draw on the Canvas:** Finally, you can draw on the canvas. Set the fill style of the drawing object to the color red:

  ```
  ctx.fillStyle = "red";
  ```

  The fillStyle property can be a CSS color, a gradient, or a pattern. The default fillStyle is black.

  The fillRect(*x,y,width,height*) method draws a rectangle, filled with the fill style, on the canvas. This means: Start at the upper-left corner (0,0) and draw a 150x75 pixels rectangle.

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

## Canvas Coordinates

The HTML canvas is a two-dimensional grid. The upper-left corner of the canvas has the coordinates (0,0)

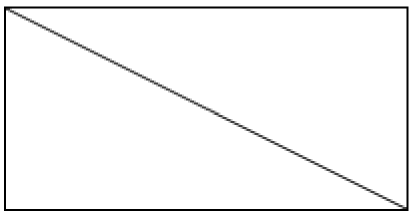
## Draw a Line:

To draw a straight line on a canvas, use the following methods:

- **moveTo(*x,y*)** - defines the starting point of the line
- **lineTo(*x,y*)** - defines the ending point of the line

- To actually draw the line, you must use one of the "ink" methods, like **stroke()**.

Define a starting point in position (0,0), and an ending point in position (200,100). Then use the stroke() method to actually draw the line:



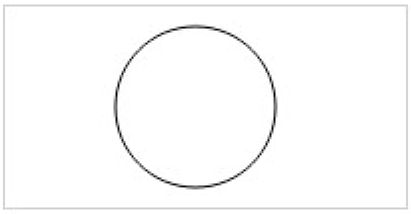
```
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid black;"></canvas>
<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
</script>
</body>
```

## Draw a Circle

To draw a circle on a canvas, use the following methods:

- **beginPath()** - begins a path
- **arc(x,y,r,startangle,endangle,direction)** - creates an arc/curve. To create a circle with arc():
  - The x and y parameters define the x- and y-coordinates of the center of the circle.
  - The r parameter defines the radius of the circle.
  - Set start angle to 0 (in radians) and end angle to 2*Math.PI (in radians).
  - direction : (optional) Boolean. True for anticlockwise , False ( default ) for clockwise direction of arc

Define a circle with the arc() method. Then use the stroke() method to actually draw the circle:



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

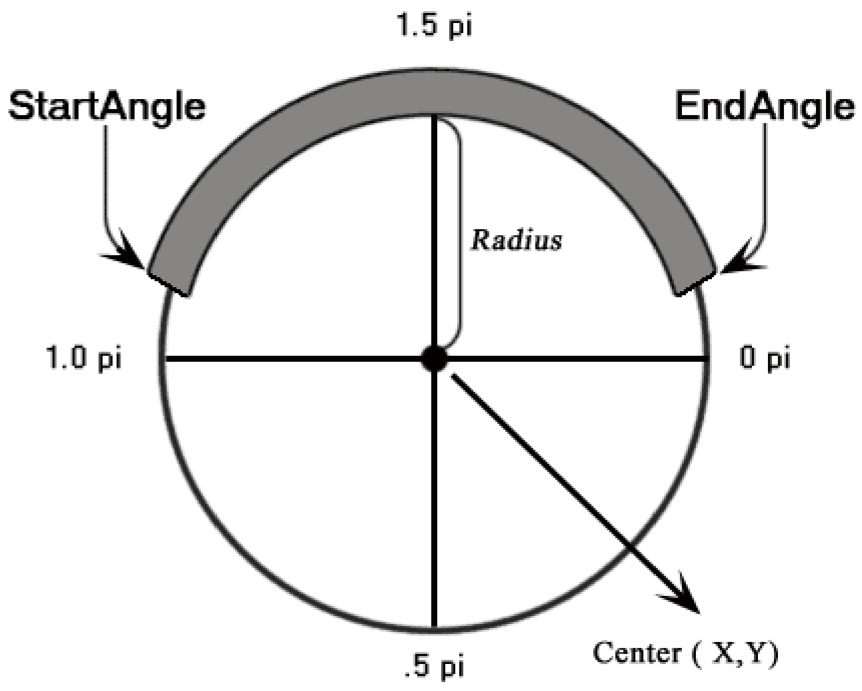
For better understanding here's a brief overview on radians:

https://www.w3resource.com/html5-canvas/html5-canvas-arc.php

Use the following formula to convert degrees to radians:

```
var radians = degrees * Math.PI/180
```

**arc Method: arc(x, y, radius, startAngle, endAngle, direction)**



**For more examples:**

https://www.plus2net.com/html_tutorial/html-canvas-arc.php

## Canvas Gradients

https://www.w3schools.com/graphics/canvas_gradients.asp

Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors.

There are two different types of gradients:

- createLinearGradient(*x,y,x1,y1*) - creates a linear gradient
- createRadialGradient(*x,y,r,x1,y1,r1*) - creates a radial/circular gradient

Once we have a gradient object, we must add two or more color stops.

**addColorStop()** method : The **addColorStop()** method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.

```
gradient.addColorStop(stop,color);
```

**fillStyle or strokeStyle property :** To use the gradient, set the **fillStyle or strokeStyle property** to the gradient, then draw the shape (rectangle, text, or a line).

The fillStyle property sets or returns the color, gradient, or pattern used to fill the drawing.

```
context.fillStyle=color|gradient|pattern;
```

The strokeStyle property sets or returns the color, gradient, or pattern used for strokes.

```
context.strokeStyle=color|gradient|pattern;
```

## Using createLinearGradient()

createLinearGradient(*x*,*y*,*x1*,*y1*) - create a linear gradient. Fill rectangle with the gradient:

Define a gradient (left to right) that goes from red to white, as the fill style for the rectangle:

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

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createLinearGradient(0, 0, 200, 0);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
</ script>
</ body>
```

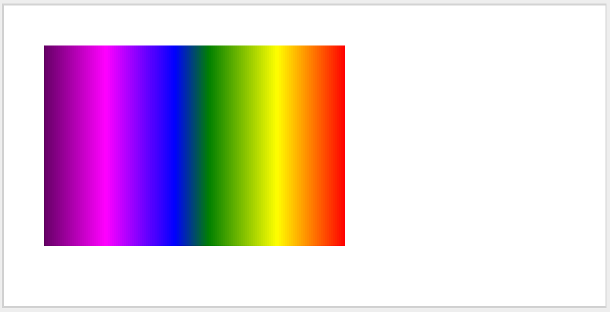
Define a gradient (top to bottom) as the fill style for the rectangle:

change the coordinates as below:

```
var grd = ctx.createLinearGradient(0, 0, 0, 200);
```

Define a gradient with multiple addColorStop() methods, as the fill style for the rectangle:

```
var grd = ctx.createLinearGradient(0, 0, 170, 0);
grd.addColorStop(0, "black");
grd.addColorStop("0.3", "magenta");
grd.addColorStop("0.5", "blue");
grd.addColorStop("0.6", "green");
grd.addColorStop("0.8", "yellow");
grd.addColorStop(1, "red");
```

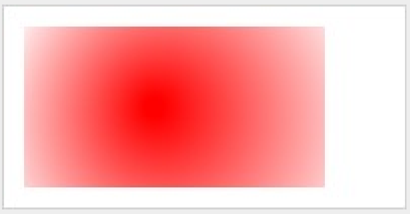


## Using createRadialGradient():

createRadialGradient(*x,y,r,x1,y1,r1*) - create a radial/circular gradient. Fill rectangle with the gradient:

JavaScript:

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
</ script>
```



## Canvas Text

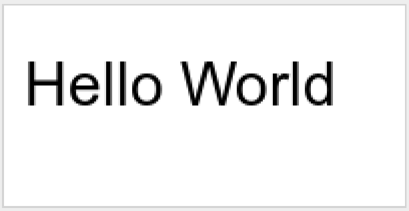
To draw text on a canvas, the most important property and methods are:

- **font** - defines the font properties for the text
- **fillText(*text,x,y*)** - draws "filled" text on the canvas
- **strokeText(*text,x,y*)** - draws text on the canvas (no fill)

### Using fillText()

Set font to 30px "Arial" and write a filled text on the canvas:

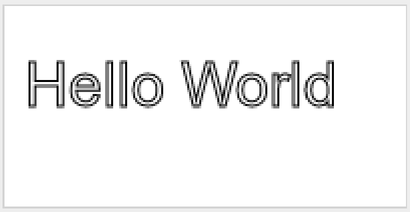
```
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid black;"></canvas>

<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
</ script>
</ body>
```



### Using strokeText()

Set font to 30px "Arial" and write a text, with no fill, on the canvas:

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World", 10, 50);
```

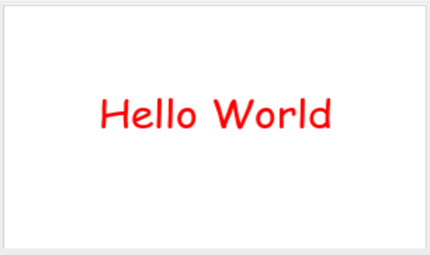


#### Add Color and Center Text

Set font to 30px "Comic Sans MS" and write a filled red text in the center of the canvas:

JavaScript:

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Comic Sans MS";
ctx.fillStyle = "red";
ctx.textAlign = "center";
ctx.fillText("Hello World", canvas.width/2, canvas.height/2);
```



## HTML5 Audio

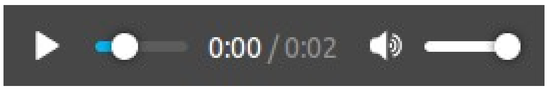
Before HTML5, audio files could only be played in a browser with a plug-in (like flash).

The HTML5 `<audio>` element specifies a standard way to embed audio in a web page.

Only MP3, WAV, and Ogg audio are supported by the HTML5 standard.

### The HTML <audio> Element

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



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

**Explanation:**

- <audio> element defines sound content
- 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.

### HTML Audio - Media Types:

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

## HTML5 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.

Only MP4, WebM, and Ogg video are supported by the HTML5 standard.

### The HTML <video> Element

To show a video in HTML, use the `<video>` element:

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

- <video> element defines a video or movie
- 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.

### HTML Video - Media Types:

| File Format | Media Type |
|---|---|
| MP4 | video/mp4 |
| WebM | video/webm |
| Ogg | video/ogg |

### HTML <video> Autoplay:

To start a video automatically use the `autoplay` attribute:

```
<video width="320" height="240" autoplay>
```

## Web Storage

https://www.w3schools.com/html/html5_webstorage.asp

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.

**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..
}
```

## The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

```
<body>
<script>
if (typeof(Storage) !== "undefined") {
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
} else {
document.getElementById("result").innerHTML = "Sorry, your browser does not support Web
Storage...";
}
</ script>
</ body>
```

Example explained:

- Create a localStorage name/value pair with name="lastname" and value="Smith"
- Retrieve the value of "lastname" and insert it into the element with id="result"

**Note:** Name/value pairs are always stored as strings. Remember to convert them to another format when needed!

The following example counts the number of times a user has clicked a button. In this code the value string is converted to a number to be able to increase the counter. Close the browser tab (or window), and try again, and the counter will continue to count (is not reset).

```
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).";
```

## The 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. Close the browser tab (or window), and try again, and the counter is reset.

```
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.";
```

## Geolocation and GPS Services

https://www.w3schools.com/html/html5_geolocation.asp

## Web workers

https://www.w3schools.com/html/html5_webworkers.asp

## Difference between HTML and HTML5?

| HTML | HTML5 |
|---|---|
| It didn’t support audio and video without the use of flash player support. | It supports audio and video controls with the use of <audio> and <video> tags. |
| It uses cookies to store  application data. | With web storage, web applications can store data locally within the user's browser. |
| Does not allow JavaScript to run in browser. | Allows JavaScript to run in background. This is possible due to JS Web worker API in HTML5. |
| Vector graphics is possible in HTML with the help of various technologies such as VML, Silver-light, Flash, etc. | Vector graphics is additionally an integral a part of HTML5 like SVG and canvas. |
| It does not allow drag and drop effects. | It allows drag and drop effects. |
| Not possible to draw shapes like circle, rectangle, triangle etc. | HTML5 allows to draw shapes like circle, rectangle, triangle etc. |
| It works with all old browsers. | It supported by all new browser like Firefox, Mozilla, Chrome, Safari, etc. |
| Older version of HTML are less mobile-friendly. | HTML5 language is more mobile-friendly. |
| Doctype declaration is too long and complicated. | Doctype declaration is quite simple and easy. |
| Elements like nav, header were not present. | New element for web structure like nav, header, footer etc. |
| Character encoding is long and complicated. | Character encoding is simple and easy. |
| It is almost impossible to get true GeoLocation of user with the help of browser. | One can track the GeoLocation of a user easily by using JS GeoLocation API. |
| It can not handle inaccurate syntax. | It is capable of handling inaccurate syntax. |
| Attributes like charset, async and ping are absent in HTML. | Attributes of charset, async and ping are a part of HTML 5. |

## New Markup Elements in  HTML5

https://www.w3schools.com/html/html5_semantic_elements.asp

## Forms

### Input type element introduced in HTML5

https://www.w3schools.com/tags/att_input_type.asp

https://www.w3schools.com/html/html_form_input_types.asp

HTML5 has the following new input types:

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

### New Input Attributes introduced in HTML5

https://www.w3schools.com/html/html_form_attributes.asp

min and max
multiple
pattern
placeholder
required
step
autofocus
height and width
list
autocomplete