HTML5 Tags

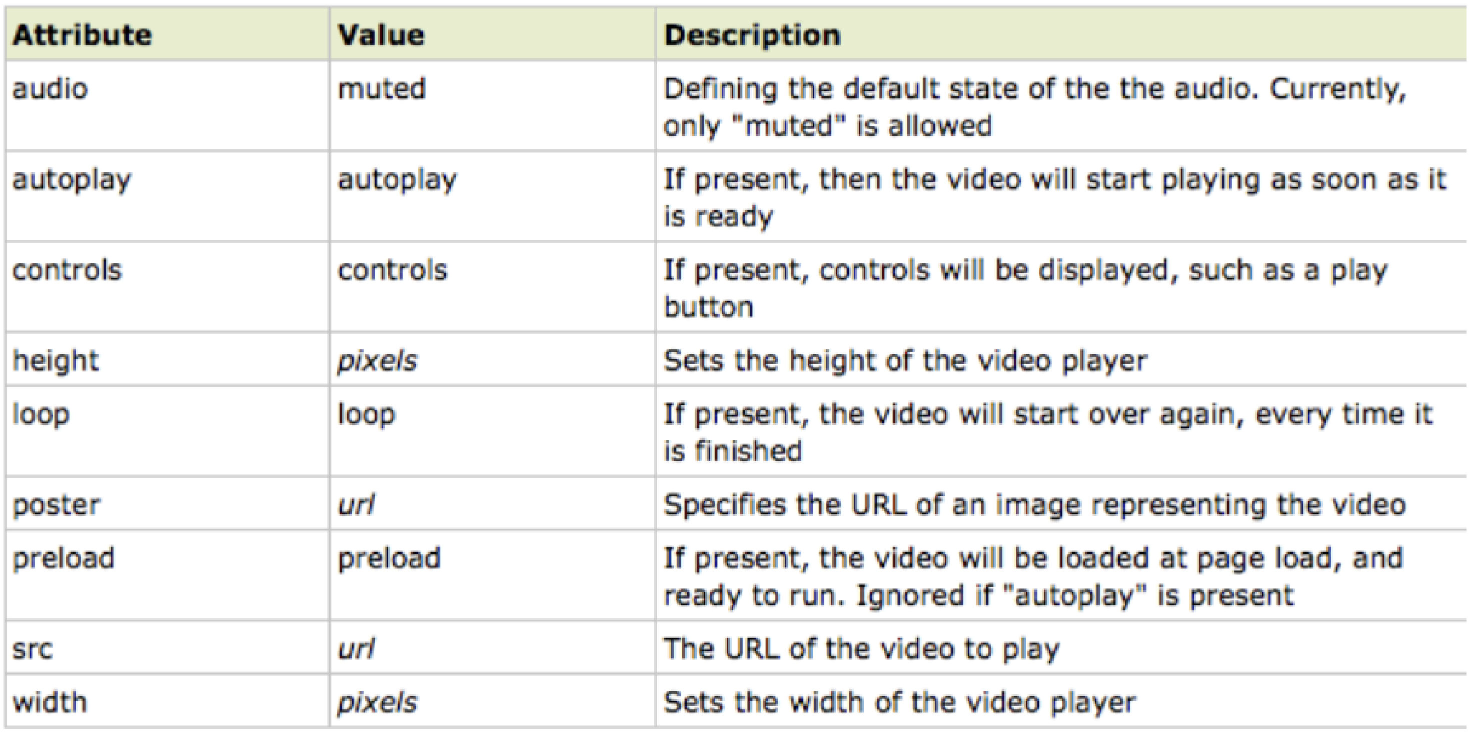
Audio

```
<audio src="my_music.mp3" controls></audio>
```

- autoplay
- buffered
- controls: if present, controls are displayed
- loop
- muted
- preload: loaded at page load, ready to play. Ignored if autoplay is present
- src
- volume

Video

```
<video src="vid.mp3" controls width="170" height="85">
```



Attribute	Value	Description
audio	muted	Defining the default state of the the audio. Currently, only "muted" is allowed
autoplay	autoplay	If present, then the video will start playing as soon as it is ready
controls	controls	If present, controls will be displayed, such as a play button
height	*pixels*	Sets the height of the video player
loop	loop	If present, the video will start over again, every time it is finished
poster	*url*	Specifies the URL of an image representing the video
preload	preload	If present, the video will be loaded at page load, and ready to run. Ignored if "autoplay" is present
src	*url*	The URL of the video to play
width	*pixels*	Sets the width of the video player

Progress

```
<progress value="33" max="100"></progress>
```

Canvas API

```
<canvas width="100" height="100"></canvas>
<script>
        let canvas = document.querySelector("canvas");
        let context = canvas.getContext("2d");
        context.fillStyle = "blue";
        context.fillRect(10,10,100,100);
</script>
```

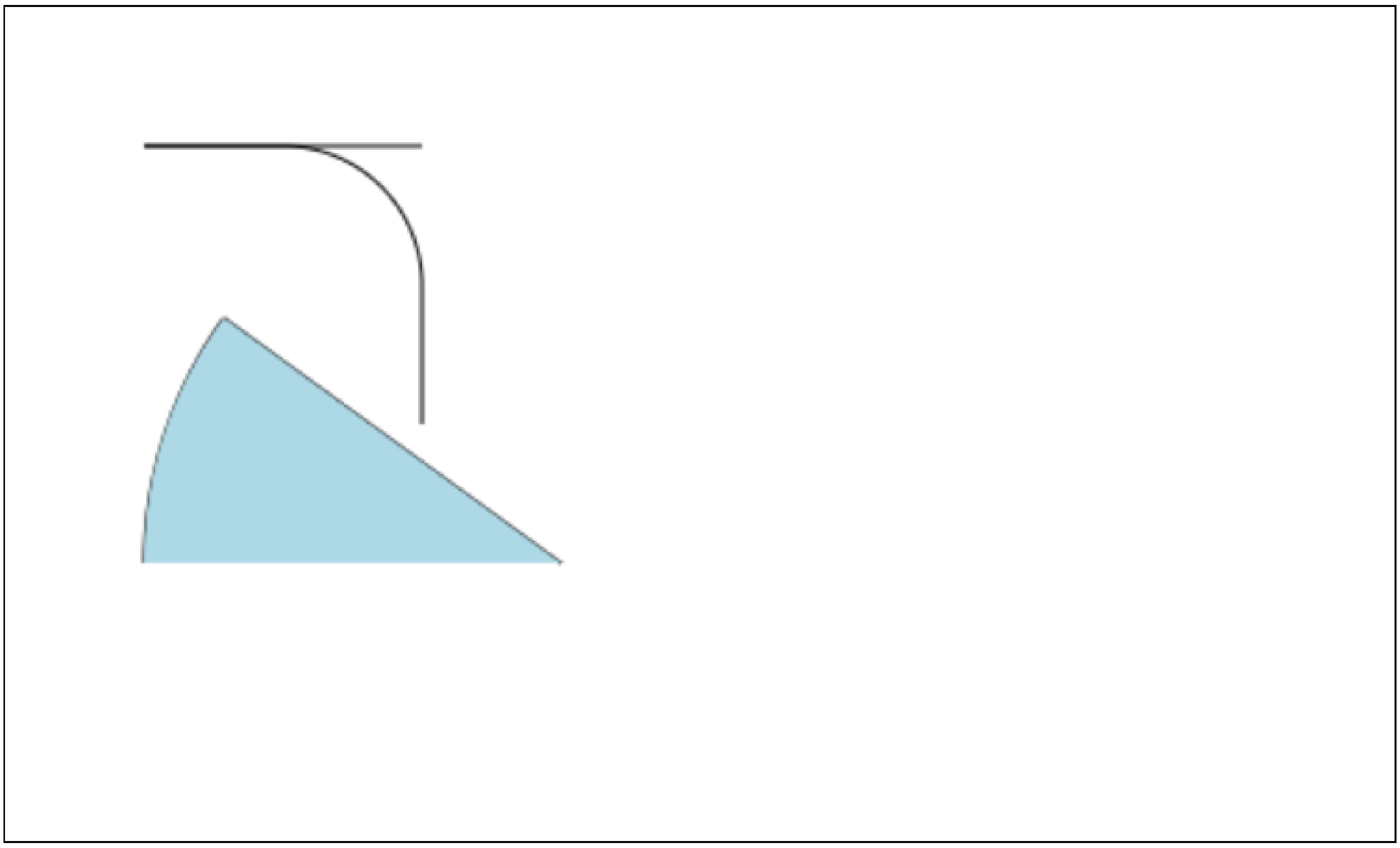
Method	Description
fillRect(x, y, width, height)	Draws a filled rectangle
strokeRect(x, y, width, height)	Draws a rectangular outline
clearRect(x, y, width, height)	Clears the specified rectangular area, making it fully transparent
moveTo(x, y)	Moves the pen to the coordinates specified by x and y
lineTo(x, y)	Draws a line from the current drawing position to the position specified by x and y
arc(x, y, r, sAngle, eAngle, anticlockwise)	Draws an arc centered at (x, y) with radius r starting at sAngle and ending at eAngle going anticlockwise (defaulting to clockwise).
arcTo(x1, y1, x2, y2, radius)	Draws an arc with the given control points and radius, connected to the previous point by a straight line

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Canvas Linear Gradient</title>
</head>
<body>

<canvas id="myCanvas" width="500" height="300" style="border:1px solid #000;">
```

```html
</canvas>

<script>
    // Get the canvas element
    var canvas = document.getElementById('myCanvas');
    var ctx = canvas.getContext('2d');

    // Create a linear gradient (startX, startY, endX, endY)
    var gradient = ctx.createLinearGradient(0, 0, canvas.width, 0);

    // Add color stops to the gradient (offset, color)
    gradient.addColorStop(0, 'blue');   // Start color
    gradient.addColorStop(1, 'red');    // End color

    // Set the fill style to the gradient
    ctx.fillStyle = gradient;

    // Fill the rectangle with the gradient
    ctx.fillRect(0, 0, canvas.width, canvas.height);
</script>

</body>
</html>
```

```javascript
const canvas = document.getElementById('myCanvas');
const ctx = canvas.getContext('2d');

// Move to the starting point
ctx.beginPath();
ctx.moveTo(50, 50);

// Draw an arc between the two lines
ctx.arcTo(150, 50, 150, 150, 50);

// Draw lines to visualize the tangents
ctx.lineTo(150, 150);
ctx.moveTo(50, 50);
ctx.lineTo(150, 50);

// Stroke the path
ctx.stroke();
ctx.beginPath();

// Draw an arc slightly larger than a semicircle
// arc(x, y, radius, startAngle, endAngle, anticlockwise)
ctx.arc(200, 200, 150, Math.PI, 1.2 * Math.PI); // Starts at 180 degrees (Math.PI), ends at 1.2 * Math.PI (~216 degrees)

// Close the path to complete the shape
ctx.lineTo(200, 200);

// Stroke the outline
```

```
ctx.stroke();

// Optionally, fill the shape
ctx.fillStyle = 'lightblue';
ctx.fill();
```



SVG API

- Scalable Vector Graphics: lossless quality regardless of scale
- Type of graphic based on XML
- `xmlns="http://www.w3.org/2000/svg"` defines that the elements inside the `<svg>` tag are part of the **SVG namespace**.

```
<svg xmlns="http://www.w3.org/2000/svg">
        <rect x="10" y="10" width="20" height="30" class="styled-rect" />
</svg>
<style>
        styled-rect{
                fill: yellow;
                stroke: black;
        }
</style>
 svg elements are also DOM elements that can be interacted with by scripts
```

Geolocation API

- Enables website to access user's location if given permission

- Accesses using `navigator.geolocation` object through a few functions:
    - getCurrentPosition
    - watchPosition
    - clearWatch

```
navigator.geolocation.getCurrentPosition(
  successCallback,
  errorCallback,
  {
    enableHighAccuracy: true,
    timeout: 5000,  // 5 seconds timeout
    maximumAge: 0   // No cached data
  }
);

function successCallback(position) {
  console.log("Latitude: " + position.coords.latitude);
  console.log("Longitude: " + position.coords.longitude);
}

function errorCallback(error) {
  console.error("Error occurred. Error code: " + error.code);
}
```

When an error occurs, the error callback function receives an error object, which can have one of the following codes:

- `1` : **PERMISSION_DENIED** – The user denied permission to access their location.
- `2` : **POSITION_UNAVAILABLE** – The location information is not available.
- `3` : **TIMEOUT** – The request for the location timed out as per the specified timeout setting.