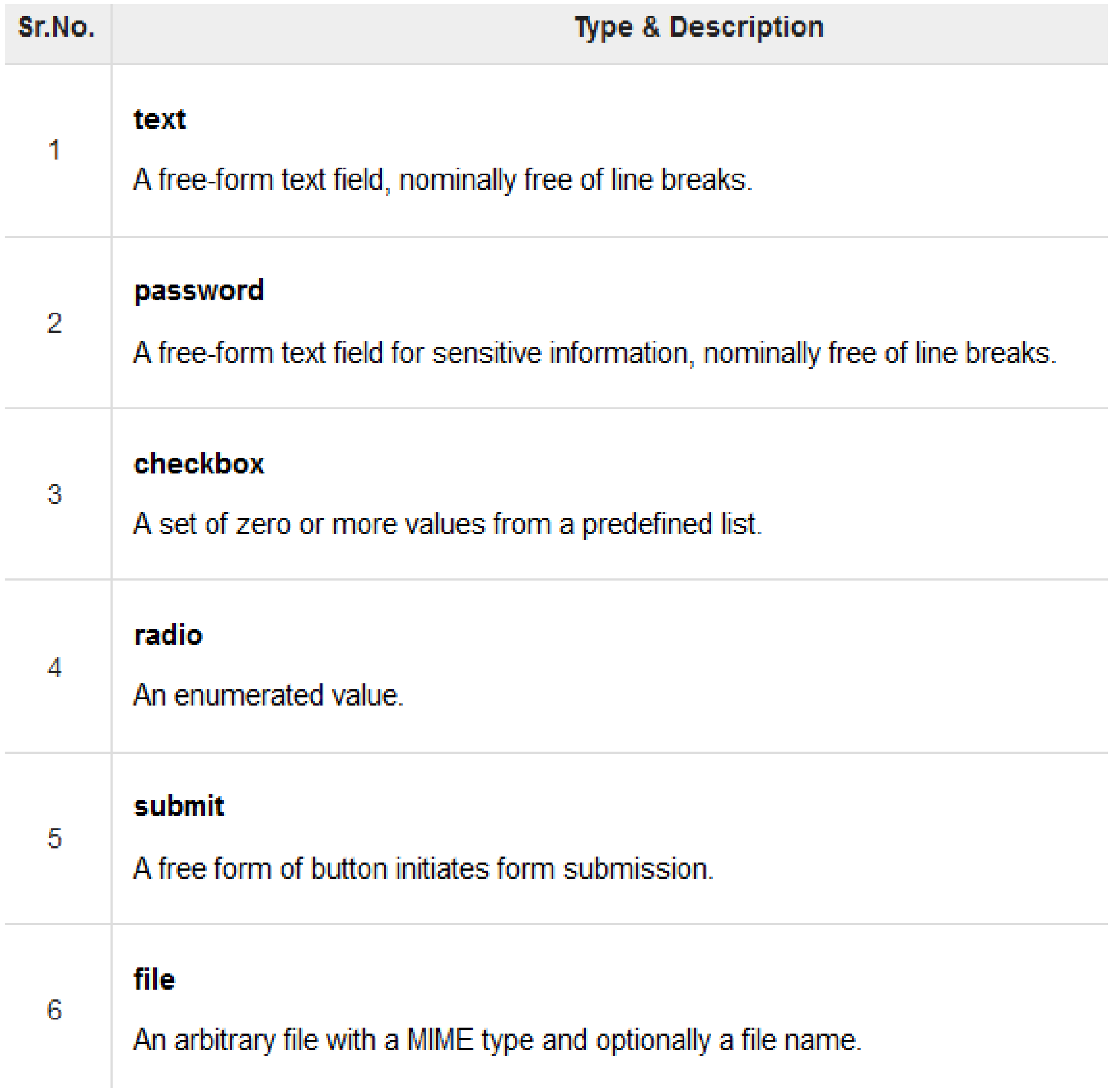
Overview of HTML5

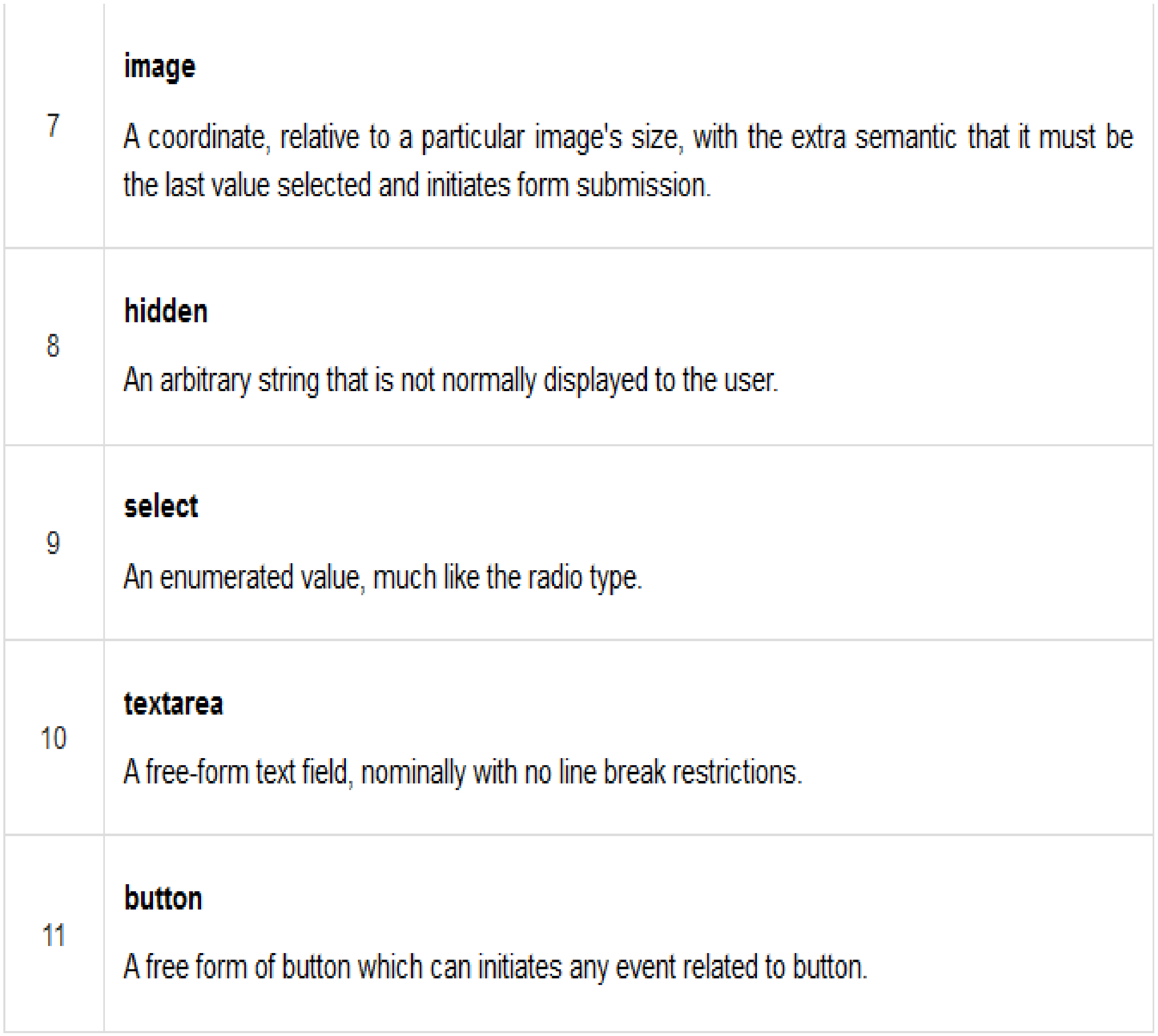
- History, Vision & Future of HTML5
- Structure of a Web Page
- HTML5 Mark-up
- Browser Support
- Forms
- Audio and Video
- Canvas
- SVG
- Geo location

The <input> element in HTML4



Sr.No.	Type & Description
1	**text** A free-form text field, nominally free of line breaks.
2	**password** A free-form text field for sensitive information, nominally free of line breaks.
3	**checkbox** A set of zero or more values from a predefined list.
4	**radio** An enumerated value.
5	**submit** A free form of button initiates form submission.
6	**file** An arbitrary file with a MIME type and optionally a file name.

The <input> element in HTML4

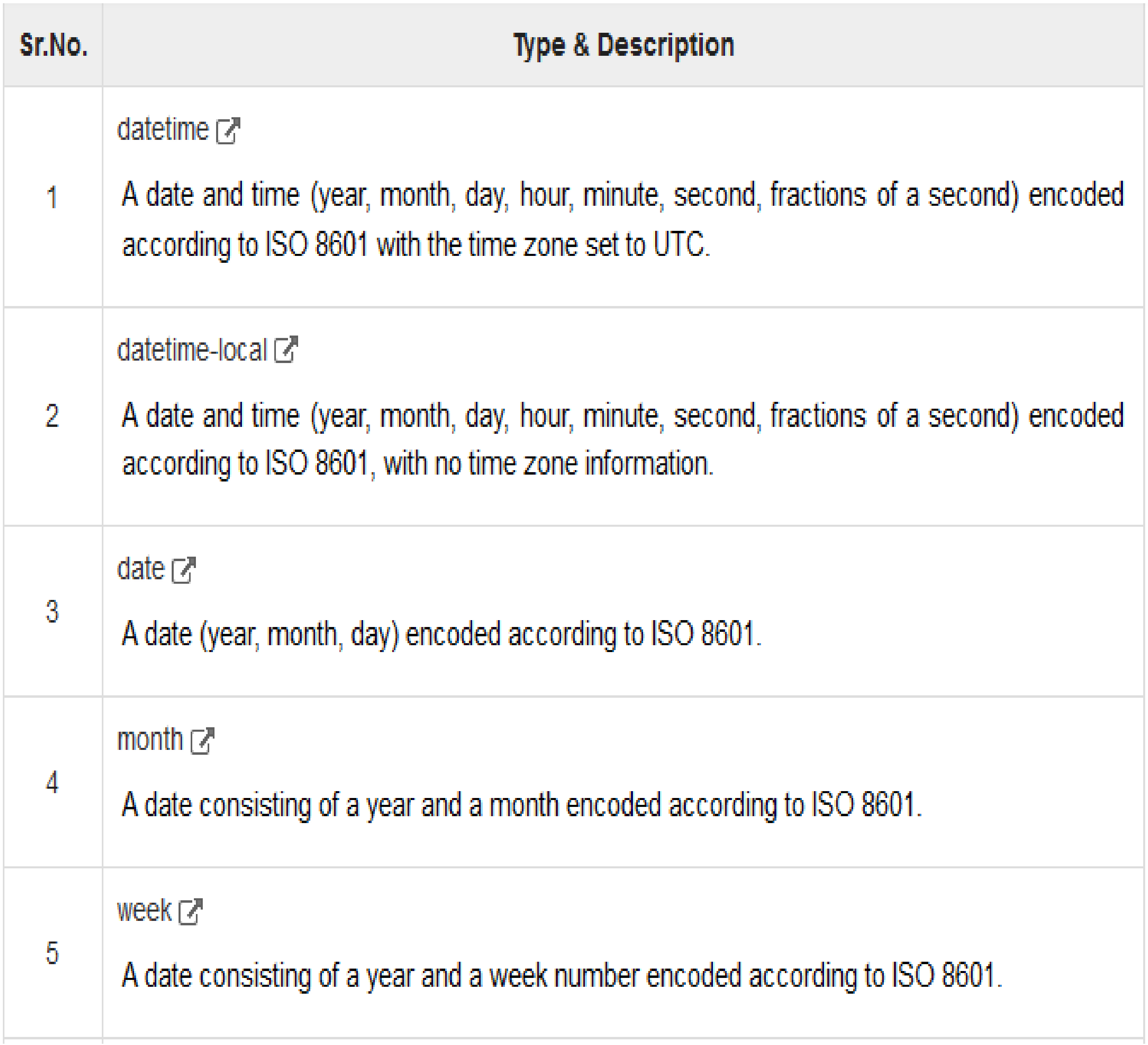


7	**image** A coordinate, relative to a particular image's size, with the extra semantic that it must be the last value selected and initiates form submission.
8	**hidden** An arbitrary string that is not normally displayed to the user.
9	**select** An enumerated value, much like the radio type.
10	**textarea** A free-form text field, nominally with no line break restrictions.
11	**button** A free form of button which can initiates any event related to button.

The <input> element in HTML4

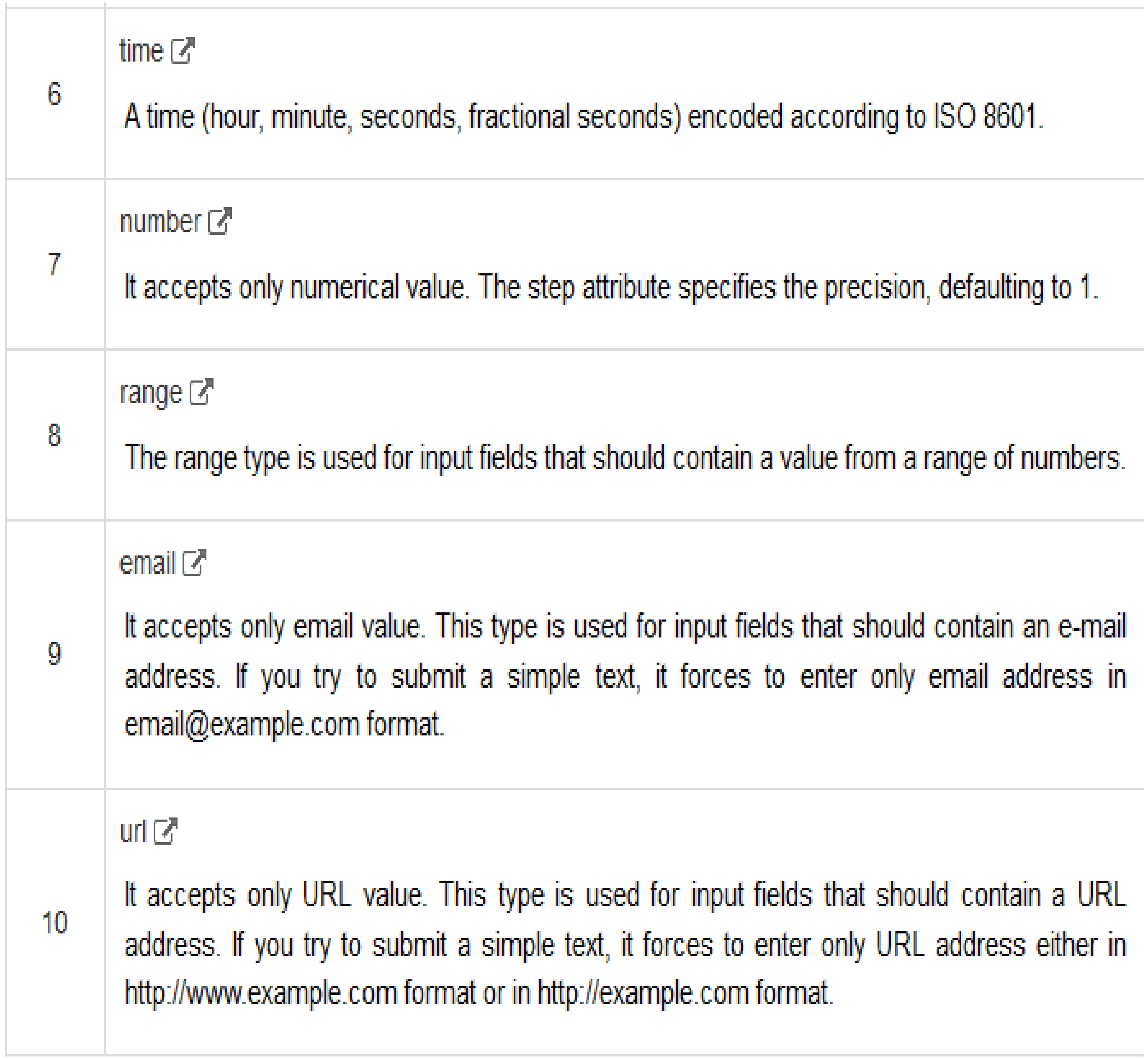
```
<html>
<form action = "http://example.com/cgiscript.pl" method = "post">
  <p>
    <label for = "firstname">first name: </label>
    <input type = "text" id = "firstname"><br />

    <label for = "lastname">last name: </label>
    <input type = "text" id = "lastname"><br />

    <label for = "email">email: </label>
    <input type = "text" id = "email"><br>

    <input type = "radio" name = "sex" value = "male"> Male<br>
    <input type = "radio" name = "sex" value = "female"> Female<br>
    <input type = "submit" value = "send"> <input type = "reset">
  </p>
</form>
</html>
```

The <input> element in HTML5



Sr.No.	Type & Description
1	datetime A date and time (year, month, day, hour, minute, second, fractions of a second) encoded according to ISO 8601 with the time zone set to UTC.
2	datetime-local A date and time (year, month, day, hour, minute, second, fractions of a second) encoded according to ISO 8601, with no time zone information.
3	date A date (year, month, day) encoded according to ISO 8601.
4	month A date consisting of a year and a month encoded according to ISO 8601.
5	week A date consisting of a year and a week number encoded according to ISO 8601.

The <input> element in HTML5



6	time A time (hour, minute, seconds, fractional seconds) encoded according to ISO 8601.
7	number It accepts only numerical value. The step attribute specifies the precision, defaulting to 1.
8	range The range type is used for input fields that should contain a value from a range of numbers.
9	email It accepts only email value. This type is used for input fields that should contain an e-mail address. If you try to submit a simple text, it forces to enter only email address in email@example.com format.
10	url It accepts only URL value. This type is used for input fields that should contain a URL address. If you try to submit a simple text, it forces to enter only URL address either in http://www.example.com format or in http://example.com format.

The <input> element in HTML5

```
<!DOCTYPE HTML>
<html>
  <head>
    <script type = "text/javascript">
        function showResult()
        {
           x = document.forms["myform"]["newinput"].value;
           document.forms["myform"]["result"].value = x;
        }
    </script>
  </head>
  <body>

    <form action = "/cgi-bin/html5.cgi" method = "get" name = "myform">
      Enter a value : <input type = "text" name = "newinput" />
      <input type = "button" value = "Result"  onclick = "showResult();" />
      <output name = "result"></output>
    </form>

  </body>
</html>
```

The <input> element in HTML5

The placeholder attribute

- HTML5 introduced a new attribute called placeholder.

- This attribute on <input> and <textarea> elements provide a hint to the user of what can be entered in the field.

- The placeholder text must not contain carriage returns or line-feeds.

Here is the simple syntax for placeholder attribute −

```
<input type = "text" name = "search" placeholder = "search the web"/>
```

The <input> element in HTML5

```
<!DOCTYPE HTML>

<html>
  <body>

    <form action = "/cgi-bin/html5.cgi" method = "get">

      Enter email :
  <input type = "email" name = "newinput”  placeholder = "email@example.com"/>
      <input type = "submit" value = "submit" />

    </form>

  </body>
</html>
```

The <input> element in HTML5

The autofocus attribute

- This is a simple one-step pattern, easily programmed in JavaScript at the time of document load, automatically focus one particular form field.

HTML5 introduced a new attribute called autofocus which would be used as follows −

```
<input type = "text" name = "search" autofocus/>
```

The <input> element in HTML5

```
<!DOCTYPE HTML>

<html>
   <body>

      <form action = "/cgi-bin/html5.cgi" method = "get">

         Enter email : <input type = "text" name = "newinput" autofocus/>
         <p>Try to submit using Submit button</p>
         <input type = "submit" value = "submit" />

      </form>

   </body>
</html>
```

The <input> element in HTML5

The required attribute

- No need to have JavaScript for client-side validations like empty text box would never be submitted because HTML5 introduced a new attribute called required

- which would be used as follows and would insist to have a value −

```
<input type = "text" name = "search" required/>
```

The <input> element in HTML5

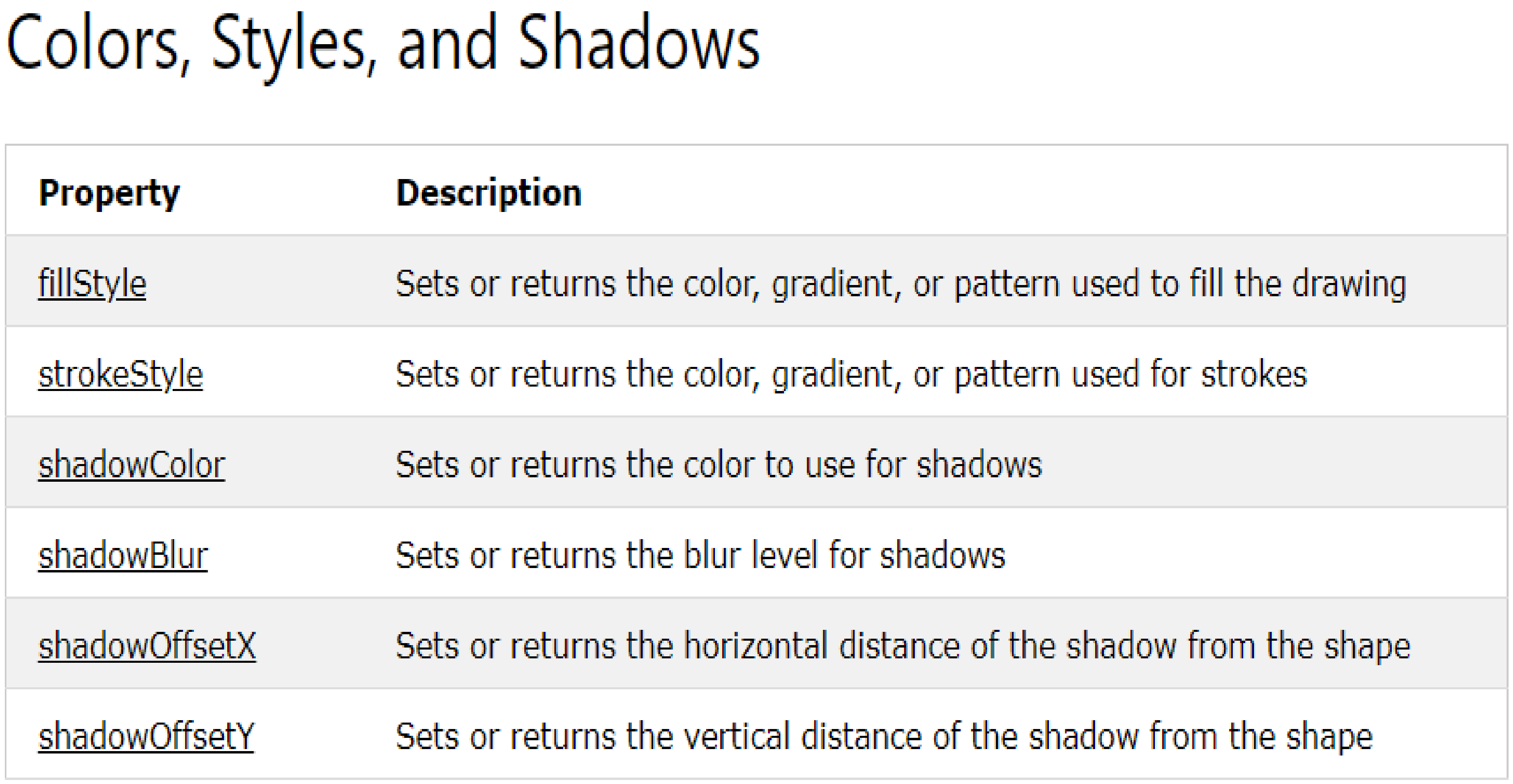
```
<!DOCTYPE HTML>

<html>
   <body>

      <form action = "/cgi-bin/html5.cgi" method = "get">

         Enter email : <input type = "text" name = "newinput" required/>
         <p>Try to submit using Submit button</p>
         <input type = "submit" value = "submit" />

      </form>

   </body>
</html>
```

HTML5 Graphics Elements

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

- <Canvas>
  - The <canvas> tag is used to draw graphics, on the fly, via scripting (usually JavaScript).
  - The <canvas> tag is only a container for graphics, you must use a script to actually draw the graphics.

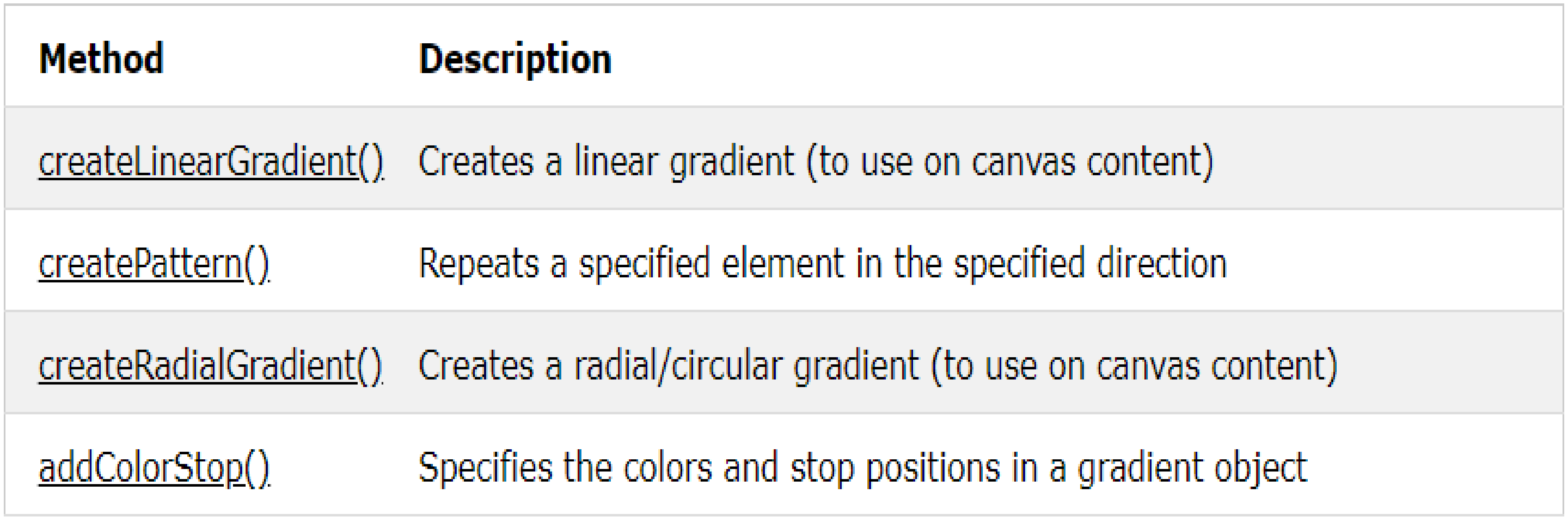
HTML5 Graphics Elements



Colors, Styles, and Shadows

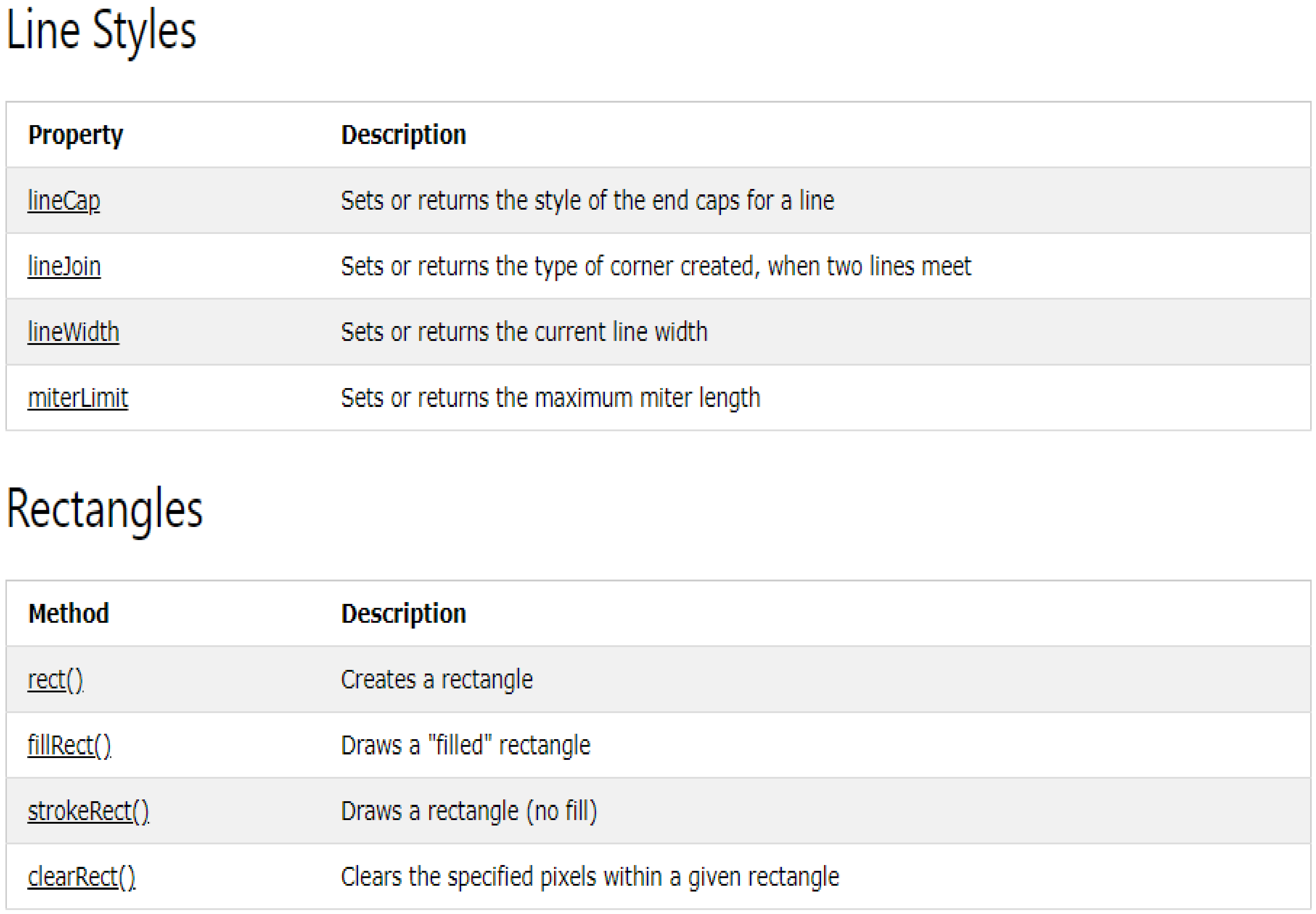
Property	Description
fillStyle	Sets or returns the color, gradient, or pattern used to fill the drawing
strokeStyle	Sets or returns the color, gradient, or pattern used for strokes
shadowColor	Sets or returns the color to use for shadows
shadowBlur	Sets or returns the blur level for shadows
shadowOffsetX	Sets or returns the horizontal distance of the shadow from the shape
shadowOffsetY	Sets or returns the vertical distance of the shadow from the shape

HTML5 Graphics Elements



Method	Description
createLinearGradient()	Creates a linear gradient (to use on canvas content)
createPattern()	Repeats a specified element in the specified direction
createRadialGradient()	Creates a radial/circular gradient (to use on canvas content)
addColorStop()	Specifies the colors and stop positions in a gradient object

HTML5 Graphics Elements



Line Styles

Property	Description
lineCap	Sets or returns the style of the end caps for a line
lineJoin	Sets or returns the type of corner created, when two lines meet
lineWidth	Sets or returns the current line width
miterLimit	Sets or returns the maximum miter length

Rectangles

Method	Description
rect()	Creates a rectangle
fillRect()	Draws a "filled" rectangle
strokeRect()	Draws a rectangle (no fill)
clearRect()	Clears the specified pixels within a given rectangle

HTML5 Graphics Elements

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

- <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.

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

By default, the <canvas> element has no border and no content.

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

HTML5 Graphics Elements

Step 1: Find the Canvas Element

First of all, you must 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");
```

HTML5 Graphics Elements

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 = "#FF0000";
```

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:

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

HTML5 Graphics Elements

Canvas Coordinates
The HTML canvas is a two-dimensional grid.

The upper-left corner of the canvas has the coordinates (0,0)

In the previous chapter, you saw this method used: fillRect(0,0,150,75).

This means: Start at the upper-left corner (0,0) and draw a 150x75 pixels rectangle.

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

HTML5 Graphics Elements

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:

```
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
```

HTML5 Graphics Elements

Draw a Circle

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

beginPath() - begins a path
arc(x,y,r,startangle,endangle) - creates an arc/curve.

To create a circle with arc( ): Set start angle to 0 and end angle to 2*Math.PI.

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.

HTML5 Graphics Elements

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

<canvas id="myCanvas" width="200" height="100“ style="border:1px solid
#d3d3d3;">
Your browser does not support the canvas element.
</canvas>

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

</body>
</html>
```

HTML5 Graphics Elements

Canvas – Gradients

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.

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

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

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

<canvas id="myCanvas" width="200" height="100“ style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</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,"white");
	grd.addColorStop(1,"red");

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

</body>
</html>
```

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

<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>

<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>

</body>
</html>
```

HTML5 Graphics Elements

Drawing Text on the Canvas

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 (x, y – positions on canvas)

strokeText(text,x,y) - draws text on the canvas (no fill)

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

<canvas id="myCanvas" width="200" height="100“ style="border:1px solid #d3d3d3;">
Your browser does not support the canvas element.
</canvas>

<script>
        var canvas = document.getElementById("myCanvas");
        var ctx = canvas.getContext("2d");
        ctx.font = "30px Arial";
        ctx.fillStyle = "red";                                          // Text color
        ctx.textAlign = "center";                                       // Text Alignment
        ctx.fillText("Hello World",10,50);
                                                              OR
        ctx.strokeText("Hello World", 10, 50);

</script>

</body>
</html>
```

HTML5 Graphics Elements

Drawing an Image on the Canvas

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

drawImage(*image*,*x*,*y*)

```
<!DOCTYPE html>
<html>
<body>
<p>Image to use:</p>

<img id="scream" width="220" height="277“ src="pic_the_scream.jpg" alt="The Scream">

<p>Canvas:</p>
<canvas id="myCanvas" width="240" height="297“ style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.
</canvas>

<script>
        window.onload = function() {
            var canvas = document.getElementById("myCanvas");
            var ctx = canvas.getContext("2d");
            var img = document.getElementById("scream");
           ctx.drawImage(img, 10, 10);
        };
</script>

</body>
</html>
```

SVG

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

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

<svg width="100" height="100">

  <circle cx="50" cy="50" r="40"
  stroke="green" stroke-width="4" fill="yellow" />

Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

SVG

SVG Code explanation:

- An SVG image begins with an <svg> element
- The width and height attributes of the <svg> element define the width and height of the SVG image
- The <circle> element is used to draw a circle
- The cx and cy attributes define the x and y coordinates of the center of the circle. If cx and cy are not set, the circle's center is set to (0, 0)
- The r attribute defines the radius of the circle
- The stroke and stroke-width attributes control how the outline of a shape appears.
- We set the outline of the circle to a 4px green "border"
- The fill attribute refers to the color inside the circle. We set the fill color to yellow
- The closing </svg> tag closes the SVG image

SVG

SVG Shapes

SVG has some predefined shape elements that can be used by developers:

- Rectangle <rect>
- Circle <circle>
- Ellipse <ellipse>
- Line <line>
- Polyline <polyline>
- Polygon <polygon>
- Path <path>

SVG

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

<svg width="400" height="100">
  <rect width="400" height="100"
  style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

SVG

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

<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:5;opacity:0.5" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

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

SVG

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

<svg height="140" width="500">
  <ellipse cx="200" cy="80" rx="100" ry="50"
style="fill:yellow;stroke:purple;stroke-width:2" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

- 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

SVG

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

<svg height="150" width="500">
  <ellipse cx="240" cy="100" rx="220" ry="30" style="fill:purple" />
  <ellipse cx="220" cy="70" rx="190" ry="20" style="fill:lime" />
  <ellipse cx="210" cy="45" rx="170" ry="15" style="fill:yellow" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

SVG

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

<svg height="210" width="500">
  <line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);stroke-width:2" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

- 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

SVG

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

<svg height="210" width="500">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;"/>
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

Audio / Video

- 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.

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

<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>

</body>
</html>
```

Audio / Video

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

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

</body>
</html>
```

HTML5 Local Storage

- Cookies are small pieces of data which a server can store in the browser.
- The cookie is sent by the browser along with all future HTTP requests to the server that set the cookie.
- Cookies cannot be bigger than 4KB in total.

HTML5 Local Storage

- HTML5 local storage is set via JavaScript executed in the browser.
- HTML5 local storage properties are never sent to any server - unless you explicitly copy them out of the local storage and appends them to an AJAX request.
- HTML5 local storage can store somewhere between 2MB and 10MB data in the browser (per origin - domain name).

- Exactly how much data is allowed depends on the browser.
- A limit of 5MB to 10MB is most common.

HTML5 Local Storage

• HTML5 local storage offers a simple **key - value store**, like a hash table or dictionary object.

• The local storage object looks very similar to a regular JavaScript object, with the exception that it is stored in the browser, even if the page is unloaded.

Object	Available to	Lifetime
localStorage	All windows or tabs using the same domain	Permanent
sessionStorage	A particular window or tab and its popups	Till the end of the session

HTML5 Local Storage

```
 <!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>
```

HTML5 Local Storage

```
<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>
```

HTML5 SessionStorage

```
<!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>
```

HTML5 SessionStorage

```
<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>
```

HTML5 Geolocation

- The Geolocation API of HTML5 helps in identifying the user’s location, which can be used to provide location specific information or route navigation details to the user.

- Check for Browser compatibility

- The geolocation property of the global navigator object helps in detecting the browser support for the Geolocation API.

```
if (navigator.geolocation)
{
    // Get the user's current position
} else
{
    alert('Geolocation is not supported in your browser');
}
```

HTML5 Geolocation

- Get the user’s current location

- The current location of the user can be obtained using the **getCurrentPosition** of the **navigator.geolocation** object.
- This function accepts three parameters – **Success** callback function, **Error** callback function and **position** options.
- If the location data is fetched successfully, the success callback function will be invoked with the obtained **position** object as its input parameter.
- Otherwise, the error callback function will be invoked with the **error** object as its input parameter.

HTML5 Geolocation

Get the user’s current location

```
 if (navigator.geolocation)
{
// Get the user's current position
navigator.geolocation.getCurrentPosition(showPosition, showError, optn);
} else
 {
		alert('Geolocation is not supported in your browser');
}
```

HTML5 Geolocation

Get the user’s current location

1. Success callback function

- This callback function is invoked only when the user accepts to share the location information and the location data is successfully fetched by the browser.
- A position object contains a **timestamp** property denoting the time at which the location data is retrieved and a **coords** object.
- The coords object contains latitude, longitude, accuracy, altitude.

2. Error
3. Position