Objective

- HTML5 Canvas
- HTML5 Inline SVG
- HTML5 Video
- HTML5 Audio

HTML5 Canvas:

- The <canvas> element is used to draw graphics, on the fly, on a web page.
- The example at the left shows a red rectangle, a gradient rectangle, a multicolor rectangle, and some multicolor text that is drawn onto the canvas.

What is Canvas?

- The HTML5 <canvas> element is used to draw graphics, on the fly, 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.

Browser Support

Internet Explorer Firefox Opera Google Chrome Safari

Create a Canvas

A canvas is a rectangular area on an HTML page, and it is specified with the <canvas> element.
Note: By default, the <canvas> element has no border and no content.
The markup looks like this:
<canvas id="myCanvas" width="200" height="100"></canvas>

To add a border, use the style attribute:
Example

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

Example

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

Your browser does not support the HTML5 canvas tag.
</canvas>
</body>
</html>

Draw Onto The Canvas With JavaScript
All drawing on the canvas must be done inside a JavaScript:

Example
<script>
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);
</script>

Example
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script>
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);
</script>
</body>
</html>

Example explained:

First, find the <canvas> element:
var c=document.getElementById("myCanvas");

Then, call its getContext() method (you must pass the string "2d" to the getContext() method):
var ctx=c.getContext("2d");

The getContext("2d") object is a built-in HTML5 object, with many properties and methods for drawing paths, boxes, circles, text, images, and more.

The next two lines draw a red rectangle:
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);

The fillStyle property can be a CSS color, a gradient, or a pattern. The default fillStyle is #000000 (black).

The fillRect(x,y,width,height) method draws a rectangle filled with the current fill style.
Canvas Coordinates

The canvas is a two-dimensional grid.

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

So, the fillRect() method above had the parameters (0,0,150,75).

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

Coordinates Example

Mouse over the rectangle below to see its x and y coordinates:

Canvas - Paths
To draw straight lines on a canvas, we will use the following two 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, we must use one of the "ink" methods, like stroke().

Example

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:

JavaScript:
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.moveTo(0,0);

ctx.lineTo(200,100);
ctx.stroke();

Example

```
<!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");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
</body>
</html>
```

To draw a circle on a canvas, we will use the following method:

arc(x,y,r,start,stop)

To actually draw the circle, we must use one of the "ink" methods, like stroke() or fill().
Example

Create a circle with the arc() method:

JavaScript:

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
</body>
</html>
```

Example

```
<!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");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
</script>
</body>
</html>
```

Canvas - Text

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

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

Using fillText():

Example

Write a 30px high filled text on the canvas, using the font "Arial":

JavaScript:

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

Example

```
<!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");
ctx.font="30px Arial";
ctx.fillText("Hello World",10,50);
</script>
</body>
</html>
```

Using strokeText():

Example
Write a 30px high text (no fill) on the canvas, using the font "Arial":

JavaScript:

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

Example

```
<!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");
ctx.font="30px Arial";
ctx.strokeText("Hello World",10,50);
</script>
</body>
</html>
```

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, and then draw the shape, like a rectangle, text, or a line.

Using createLinearGradient():

Example

Create a linear gradient. Fill rectangle with the gradient:

JavaScript:

```
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);
```

Example

```
<!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,"red");
grd.addColorStop(1,"white");
```

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

Using createRadialGradient():

Example
Create a radial/circular gradient. Fill rectangle with the gradient:

JavaScript:

```
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);
```

Example

```
<!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>
```

Canvas - Images
To draw an image on a canvas, we will use the following method:

drawImage(image,x,y)

JavaScript:
```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
var img=document.getElementById("scream");
ctx.drawImage(img,10,10);
```

Example
```
<!DOCTYPE html>
<html>
<body>
<p>Image to use:</p>
<img id="scream" src="img_the_scream.jpg" alt="The Scream" width="220"
height="277"><p>Canvas:</p>
<canvas id="myCanvas" width="250" height="300" 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");
var img=document.getElementById("scream");
ctx.drawImage(img,10,10);
</script>
</body>
</html>
```

HTML5 Inline SVG

What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define vector-based graphics for the Web
- SVG defines the graphics in XML format

- SVG graphics do NOT lose any quality if they are zoomed or resized
- Every element and every attribute in SVG files can be animated
- SVG is a W3C recommendation

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 (and the image can be zoomed without degradation)

Browser Support

Internet Explorer Firefox Opera Google Chrome Safari

In HTML5, you can embed SVG elements directly into your HTML page:

Example

```
<!DOCTYPE html>
<html>
<body>
<svg width="300" height="200">
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
</body>
</html>
```

Example

```
<!DOCTYPE html>
<html>
<body>
<svg width="300" height="200">
   <polygon points="100,10 40,180 190,60 10,60 160,180"
   style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
   Sorry, your browser does not support inline SVG.
</svg>
</body>
```

</html>

Differences Between SVG and Canvas

- SVG is a language for describing 2D graphics in XML.
- Canvas draws 2D graphics, on the fly (with a JavaScript).
- SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.
- In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.
- Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

Comparison of Canvas and SVG

The table below shows some important differences between Canvas and SVG:

Canvas

- Resolution independent
- Support for event handlers
- Best suited for applications with large rendering areas (Google Maps)
- Slow rendering if complex (anything that uses the DOM a lot will be slow)
- Not suited for game applications

SVG

- Resolution dependent
- No support for event handlers
- Poor text rendering capabilities
- You can save the resulting image as .png or .jpg
- Well suited for graphic-intensive games

HTML5 Video

- Many modern websites show videos. HTML5 provides a standard for showing them.
- Before HTML5, there was no standard for showing videos/movies on web pages.
- Before HTML5, videos could only be played with a plug-in (like flash). However, different browsers supported different plug-ins.
- HTML5 defines a new element which specifies a standard way to embed a video or movie on a web page: the <video> element.

Browser Support
Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <video> element.

Note: Internet Explorer 8 and earlier versions, do not support the <video> element.
HTML5 Video - How It Works

To show a video in HTML5, this is all you need:
Example
```
<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>
```

Example
```
<!DOCTYPE html>
<html>
<body>
<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>
</body>
</html>
```

The control attribute adds video controls, like play, pause, and volume.

It is also a good idea to always include width and height attributes. If height and width are set, the space required for the video is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the video, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the video loads).

You should also insert text content between the <video> and </video> tags for browsers that do not support the <video> element.

The <video> element allows multiple <source> elements. <source> elements can link to different video files. The browser will use the first recognized format.

HTML5 <video> - DOM Methods and Properties

HTML5 has DOM methods, properties, and events for the <video> and <audio> elements.

These methods, properties, and events allow you to manipulate <video> and <audio> elements using JavaScript.

There are methods for playing, pausing, and loading, for example and there are properties (like duration and volume). There are also DOM events that can notify you when the <video> element begins to play, is paused, is ended, etc.

The example below illustrate, in a simple way, how to address a <video> element, read and set properties, and call methods.
Example 1

Create simple play/pause + resize controls for a video:

Example

```
<!DOCTYPE html>
<html>
<body>
<div style="text-align:center">
  <button onclick="playPause()">Play/Pause</button>
  <button onclick="makeBig()">Big</button>
  <button onclick="makeSmall()">Small</button>
  <button onclick="makeNormal()">Normal</button>
  <br>
  <video id="video1" width="420">
    <source src="mov_bbb.mp4" type="video/mp4">
    <source src="mov_bbb.ogg" type="video/ogg">
    Your browser does not support HTML5 video.
    </video>
</div>
<script>
var myVideo=document.getElementById("video1");

function playPause()
{
if (myVideo.paused)
  myVideo.play();
else
  myVideo.pause();
}
```

HTML5 Audio

HTML5 provides a standard for playing audio files.

Audio on the Web
Before HTML5, there was no standard for playing audio files on a web page.
Before HTML5, audio files had to be played with a plug-in (like flash). However, different browsers supported different plug-ins.

HTML5 defines a new element which specifies a standard way to embed an audio file on a web page: the <audio> element.

Browser Support
Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <audio> element.

HTML5 Audio - How It Works

To play an audio file in HTML5, this is all you need:

Example
```
<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>
```

Example
```
<!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>
```

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

You should also insert text content between the <audio> and </audio> tags for browsers that do not support the <audio> element.

The <audio> element allows multiple <source> elements. <source> elements can link to different audio files. The browser will use the first recognized format.

Audio Formats and Browser Support

Currently, there are 3 supported file formats for the <audio> element: MP3, Wav, and Ogg:

Browser	MP3	Wav	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	NO	YES	YES
Safari	YES	YES	NO
Opera	NO	YES	YES

MIME Types for Audio Formats

Format	MIME-type
MP3	audio/mpeg
Ogg	audio/ogg
Wav	audio/wav