



Session: 17

# *Canvas and JavaScript*



# Objectives

- Describe Canvas in HTML5
- Explain the procedure to draw lines
- Explain the procedure to use color and transparency
- Explain the procedure to work with various drawing objects
- Describe working with images and text
- Describe the procedure to create Web page events with JavaScript and jQuery
- Describe the process of including external content in Web pages

# Canvas Element 1-6

The `<canvas>` element in HTML5 can be used to draw shapes on Web sites as well as to dynamically draw graphics using JavaScript.

The `<canvas>` element is represented like a rectangle on a page and allows the user to draw arcs, text, shapes, gradients, and patterns.

The `<canvas>` in HTML5 is like the `<div>`, `<table>`, or `<a>` tag except that the content used in it is rendered through JavaScript.

The `<canvas>` element does not contain any drawing abilities, instead, the drawing is done using a JavaScript code.

To make use of the `<canvas>` element, a user has to add the `<canvas>` tag on the HTML page.

Using `<canvas>` with JavaScript improves the overall performance of Web sites and avoids the requirement to download images from the sites.

# Canvas Element 2-6

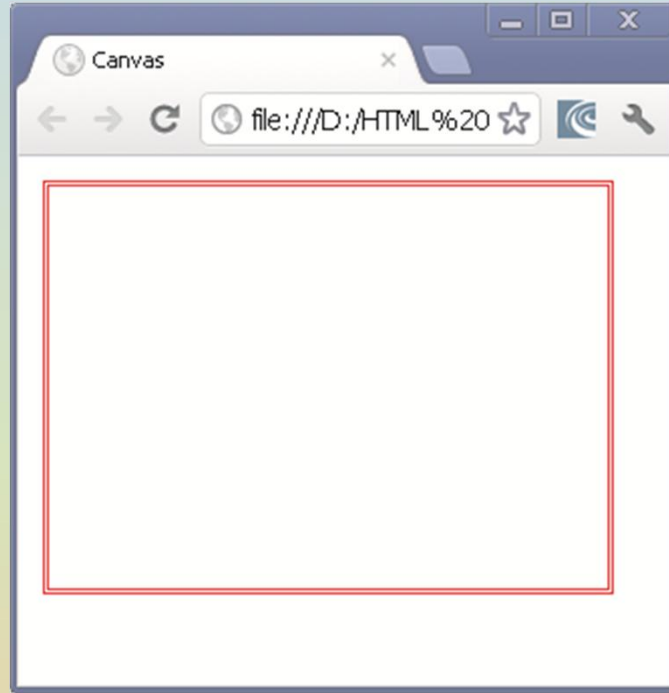
- The Code Snippet demonstrates the use of `<canvas>` element.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title> Canvas </title>
    <style>
      canvas{border: medium double red; margin: 4px}
    </style>
  </head>
  <body>
    <canvas width="278" height="200"></canvas>
  </body>
</html>
```

- In the code, the `<style>` element is used to display the border of the `<canvas>` element.
- The height and width attributes specify the size of the `<canvas>` element on the page.

# Canvas Element 3-6

- Following figure displays the `<canvas>` element.



To draw a `<canvas>` element, the user can use a context object.

The context object contains the drawing functions for a specific style of graphics.

Two-Dimensional (2d) context is used to work with 2d operations.

# Canvas Element 4-6

The `<canvas>` element in DOM exposes the `HTMLCanvasElement` interface.

This interface provides the methods and properties for changing the presentation and layout of canvas elements.

The `HTMLCanvasElement` has a `getContext(context)` method that returns the drawing context for the canvas.

- The Code Snippet demonstrates the 2d context object for the canvas.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title> Canvas </title>
    <script>
      window.onload = function()
      {
        var canvas = document.getElementById('mCanvas');
        var ctext = canvas.getContext('2d');
```

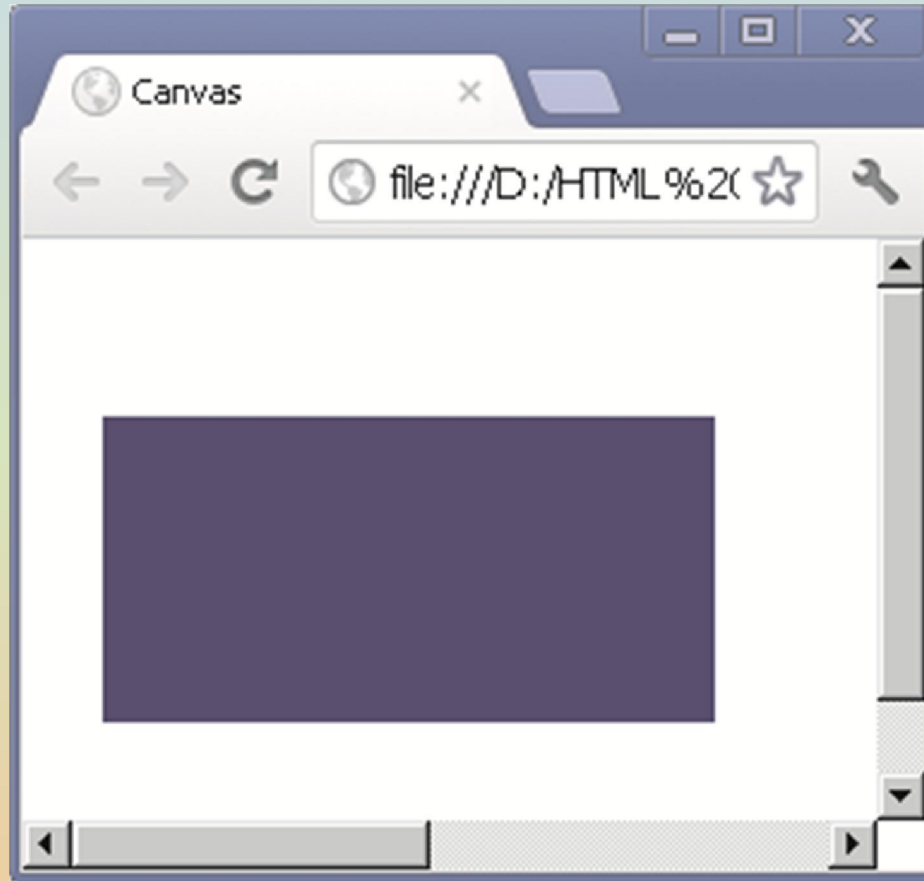
# Canvas Element 5-6

```
        ctext.beginPath();
        ctext.rect(18, 50, 200, 100);
        ctext.fillStyle = "DarkBlue";
        ctext.fill();
    };
</script>
</head>
<body>
    <canvas id="mCanvas" width="578" height="200"></canvas>
</body>
</html>
```

- In the code, the `height` and `width` attributes define the height and width of the canvas element respectively.
- In the initializer function, the DOM object is accessed through the `id` attribute and gets a 2d context by using the `getContext()` method.
- The rectangle is created by using the `rect(18, 50, 200, 100)` method with `x`, `y`, `height`, and `width` parameters and is positioned at left corner of the page.

# Canvas Element 6-6

- Following figure displays the `<canvas>` element.





# Drawing a Line in Canvas 1-4

- You can create lines in a canvas using the `stroke()`, `beginPath()`, `lineTo()`, and `moveTo()` methods.
- The following is the syntax to create a line in canvas :

## Syntax:

```
ccontext.beginPath();
```

```
ccontext.moveTo(x,y);
```

```
ccontext.lineTo(x,y);
```

```
ccontext.stroke();
```

where,

- `ccontext` - specifies a context object
- `beginPath()` - Specifies a new drawing path
- `moveTo()` - Specifies the creation of new sub path to the given position
- `lineTo()` - Specifies the drawing of a line from the context position to the given position
- `stroke()` - Specifies how to assign a color to the line and display it

# Drawing a Line in Canvas 2-4

- The Code Snippet demonstrates creating a line in HTML5 canvas.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Line</title>
    <style>
      body {
        margin: 0px;
        padding: 0px;
      }
    </style>
  </head>
  <body>
    <canvas id="mCanvas" width="360" height="200"></canvas>
  </body>
</html>

<script>
  window.onload = function() {
    var canvas = document.getElementById("mCanvas");
    var ctext = canvas.getContext("2d");

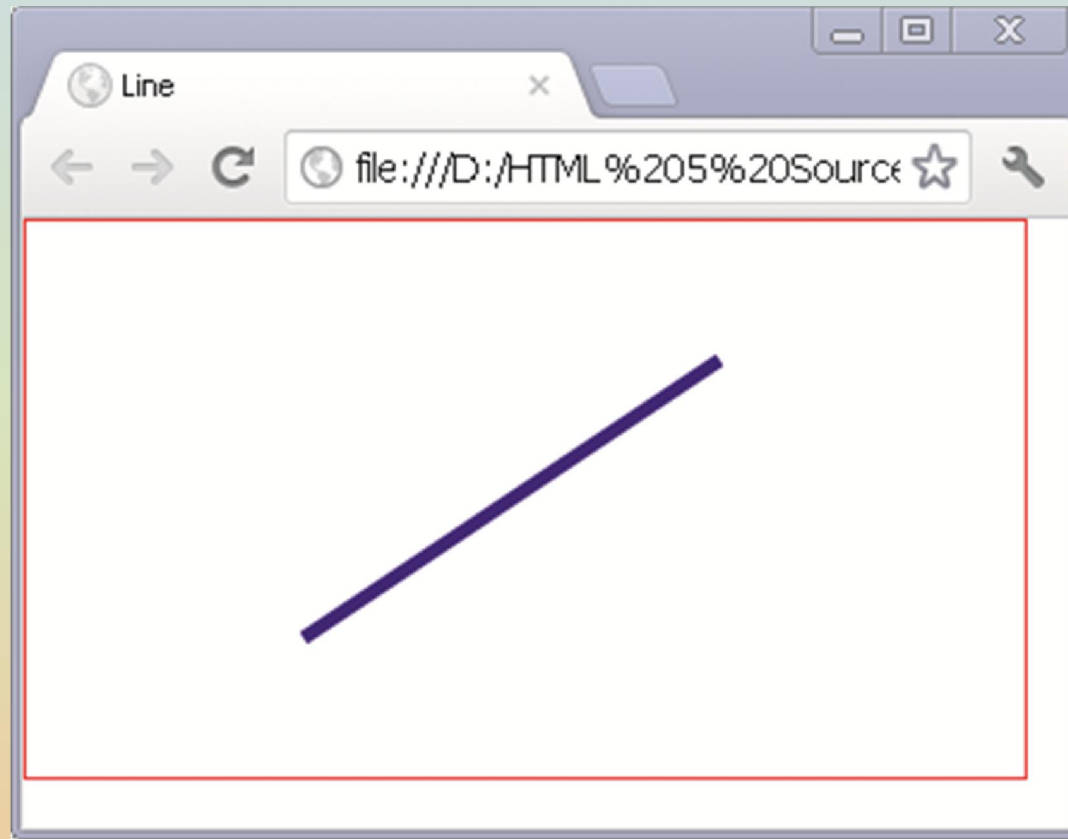
    ctext.beginPath();
    ctext.moveTo(100, 150);
    ctext.lineTo(250, 50);
    ctext.lineWidth = 5;
    ctext.strokeStyle = "blue";
    ctext.stroke();
  }
</script>
```

# Drawing a Line in Canvas 3-4

- In the code, the `height` and `width` attributes are defined.
- The initializer function has the DOM object which is accessed through the `id` attribute and gets a 2d context by using the `getContext ()` method.
- The `beginPath ()` method is called through the context object to draw the path of the line.
- The `moveTo (100, 150)` method is called that creates a new path for the given point to place the drawing cursor and moves the position of the window to the upper-left corner by giving the x and y coordinates.
- The `lineTo (250, 50)` method is called to draw the line from the context point to given point.
- The `lineWidth` property is specified as 5 to define the width of the line on the canvas.
- The `strokeStyle` property sets the color of the line to blue.
- The `stroke ()` method assigns the color to the line.

# Drawing a Line in Canvas 4-4

- Following figure displays a line drawn in a canvas.



# Working with Drawing Objects 1-17

- HTML5 canvas allows the user to work with different types of drawing objects.
- Following objects can be drawn on a canvas element:

## ➤ Rectangle

- With HTML5 canvas, the user can create a rectangle using the `rect()` method.
- The HTML5 canvas is placed by using the `x` and `y` parameters and appropriately sized through height and width properties.
- Following table lists the common properties and methods of various shapes.

Properties and Methods	Description
<code>fillStyle</code>	The values can be gradient, pattern, or a CSS color. The default property style is solid black, but the user can set the color according to the requirements.
<code>fillRect(x, y, width, height)</code>	Enables the user to draw a rectangle with the existing fill style.
<code>strokeStyle()</code>	The values can be gradient, pattern, or a CSS color.

Properties and Methods	Description
<code>strokeRect(x, y, width, height)</code>	Enables the user to draw a rectangle with the existing stroke style. This property is used to draw the edges of the rectangle.
<code>clearRect(x, y, width, height)</code>	Used to clear the pixels in a rectangle.

- The Code Snippet demonstrates how to create a rectangle in HTML5 canvas.

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      #mCanvas {
        border: 1px solid green;
      }
      body {
        margin: 0px;
        padding: 0px;
      }
    </style>
```

# Working with Drawing Objects 3-17

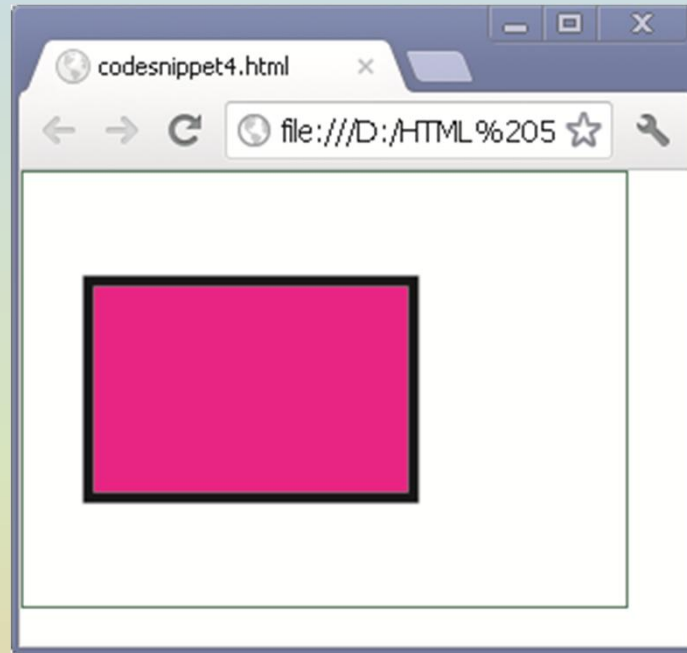
```
<script>
    window.onload = function() {
        var canvas = document.getElementById('mCanvas');
        var ctext = canvas.getContext('2d');
        ctext.beginPath();
        ctext.rect(30, 50, 150, 100);
        ctext.fillStyle = "Magenta";
        ctext.fill();
        ctext.lineWidth = 5;
        ctext.strokeStyle = 'black';
        ctext.stroke();
    };
</script>
</head>
<body>
    <canvas id="mCanvas" width="278" height="200"></canvas>
</body>
</html>
```

# Working with Drawing Objects 4-17

- In the code, the `height` and `width` attributes are defined.
- The initializer function has the DOM object which is accessed through the `id` attribute and gets a 2d context by using the `getContext()` method.
- The `beginPath()` method is called through the context object to draw the rectangle.
- The `rect(30, 50, 150, 100)` method takes `x`, `y`, `height`, and `width` as the parameters.
- The `fillStyle` property fills the rectangle with magenta color.
- The `fill()` method is used to paint the rectangle.
- The `lineWidth` property is specified as 5 to define the width of line on the canvas.
- The `strokeStyle` property sets the stroke style of the rectangle to black.
- The `stroke()` method assigns the color to the rectangle.



- Following figure displays a rectangle drawn on the canvas.



## ➤ Arcs

- With HTML5 canvas, the user can create an arc by using the `arc()` method.
- Arcs are represented using a start angle, an end angle, a radius, a center point, and the drawing direction (anticlockwise or clockwise).

- The syntax to draw an arc in HTML5 is as follows:

## Syntax:

```
arc(x, y, radius, startAngle, endAngle, anticlockwise)
```

where,

- `x, y` - Specifies the coordinates of the center of an arc
  - `radius` - Specifies the distance from the center to any point on the circle
  - `startAngle, endAngle` - Specifies the start and end points in the arc
  - `anticlockwise` - Draws the arc clockwise or anticlockwise and accepts a boolean value
- The Code Snippet demonstrates how to create an arc in HTML5 canvas.

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body
      {
        margin: 0px;
        padding: 0px;
      }
    </style>
  </head>
  <body>
```

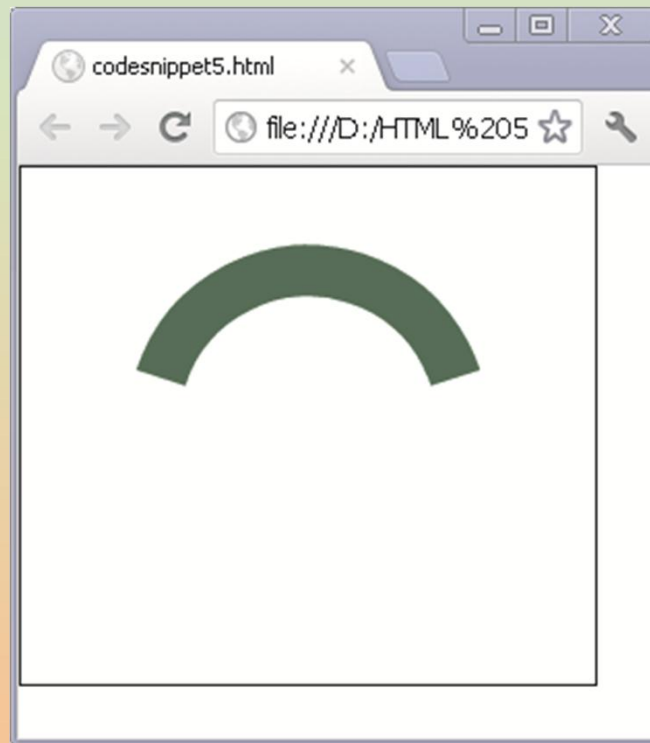


# Working with Drawing Objects 7-17

```
#mCanvas {
    border: 1px solid black; }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("mCanvas");
        var ctext = canvas.getContext("2d");
        var x = canvas.width / 2;
        var radius = 75;
        var startAngle = 1.1 * Math.PI;
        var endAngle = 1.9 * Math.PI;
        var ctrClockwise = false;
        ctext.beginPath();
        ctext.arc(x, y, radius, startAngle, endAngle, ctrClockwise);
        ctext.lineWidth = 25;
        // line color
        ctext.strokeStyle = "DarkGreen";
        ctext.stroke();
    };
</script> </head>
<body>
    <canvas id="mCanvas" width="278" height="250"></canvas>
</body></html>
```

# Working with Drawing Objects 8-17

- In the code, the `beginPath()` method is called through the context object to draw an arc by using the `arc()` method which has `x`, `y`, and `radius` as the parameters.
- The `startAngle` and the `endAngle` are the start and end points of the arc.
- The `anticlockwise` specifies the direction of the arc between the two start and end points.
- Following figure displays an arc in HTML5 canvas.



## ➤ Circle

- In HTML5, you can draw a circle using the `arc()` method.
- You have to set the start angle with 0 and the end angle is specified as  $2 * \text{PI}$ .
- Following is the syntax to draw a circle in HTML5 is as follows:

### Syntax:

```
arc(x, y, radius, startAngle, endAngle, anticlockwise)
```

where,

- `x, y` - Specifies the coordinates of the center of an arc
- `radius` - Specifies the distance from the center to any point on the circle
- `startAngle, endAngle` - Specifies the start and end points in the arc
- `anticlockwise` - Draws the arc clockwise or anticlockwise and accepts a boolean value

# Working with Drawing Objects 10-17

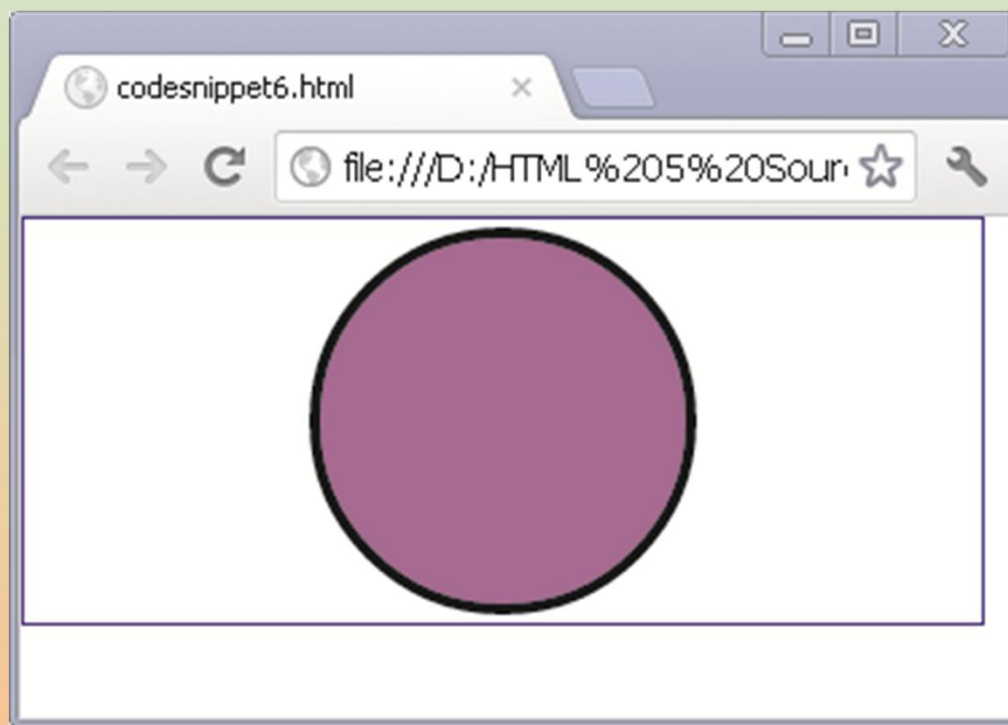
- The Code Snippet demonstrates how to create a circle using HTML5.

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      var canvas = document.getElementById("mCanvas");
      var ctx = canvas.getContext("2d");
      var radius = 70;
      var ctrX = canvas.width / 2;
      var ctrY = canvas.height / 2;
      ctx.beginPath();
      ctx.arc(ctrX, ctrY, radius, 0, 2 * Math.PI, false);
      ctx.fillStyle = "DarkOrchid";
      ctx.fill();
      ctx.strokeStyle = "black";
      ctx.stroke();
    </script>
  </head>
  <body>
    <canvas id="mCanvas" width="356" height="150"></canvas>
  </body>
</html>

window.onload = function() {
  var canvas = document.getElementById("mCanvas");
  var ctx = canvas.getContext("2d");
  var ctrX = canvas.width / 2;
```

# Working with Drawing Objects 11-17

- In this code, a circle is defined by using the `arc()` method which has `ctrX`, `ctrY`, and `radius` as the parameters.
- To define the arc with the points the `startAngle` is set to 0 and the `endAngle` is specified as  $2 * \text{PI}$ .
- The `anticlockwise` defines the direction of the path of an arc between the two start and end points.
- Following figure displays a circle in HTML5 canvas.



## ➤ Bezier Curves

- Using HTML5 canvas, you can create a Bezier curve using the `bezierCurveTo()` method.
- Bezier curves are represented with the two control points, context points, and an end point.
- The Code Snippet demonstrates how to create a Bezier curve using HTML5.

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body
      {
        margin: 0px;
        padding: 0px;
      }
      #mCanvas
      {
        border: 1px solid maroon;
      }
    </style>
```

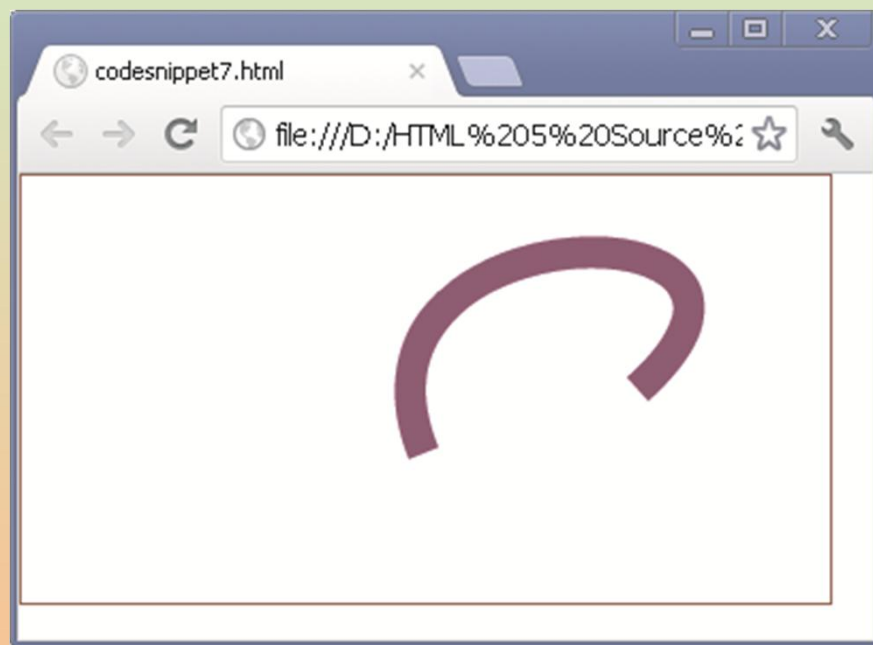


```
<script>
    window.onload = function()
    {
        var canvas = document.getElementById("mCanvas");
        var ctext = canvas.getContext("2d");
        ctext.beginPath();
        ctext.moveTo(188, 130);
        ctext.bezierCurveTo(140, 10, 388, 10, 288, 100);
        ctext.lineWidth = 15;
        // line color
        ctext.strokeStyle = "purple";
        ctext.stroke();
    };
</script>
</head>
<body>
    <canvas id="mCanvas" width="378" height="200"></canvas>
</body>
</html>
```



# Working with Drawing Objects 14-17

- In this code, the Bezier curve uses the `bezierCurveTo()` method.
- This method defines the current context point, two control points, and an end point.
- The context point uses the `moveTo()` method.
- The first portion of the curve is tangential to the imaginary line defined in the context point and first control point.
- The second portion of the curve is tangential to the imaginary line which is defined by the second control point and the ending point.
- Following figure displays a Bezier curve in a canvas.



## ➤ Quadratic Curves

- HTML5 canvas allows the user to create quadratic curves using the `quadraticCurveTo()` method.
- Quadratic curves are represented through the context point, an end point, and a control point.
- The Code Snippet demonstrates how to create a quadratic curve using HTML5.

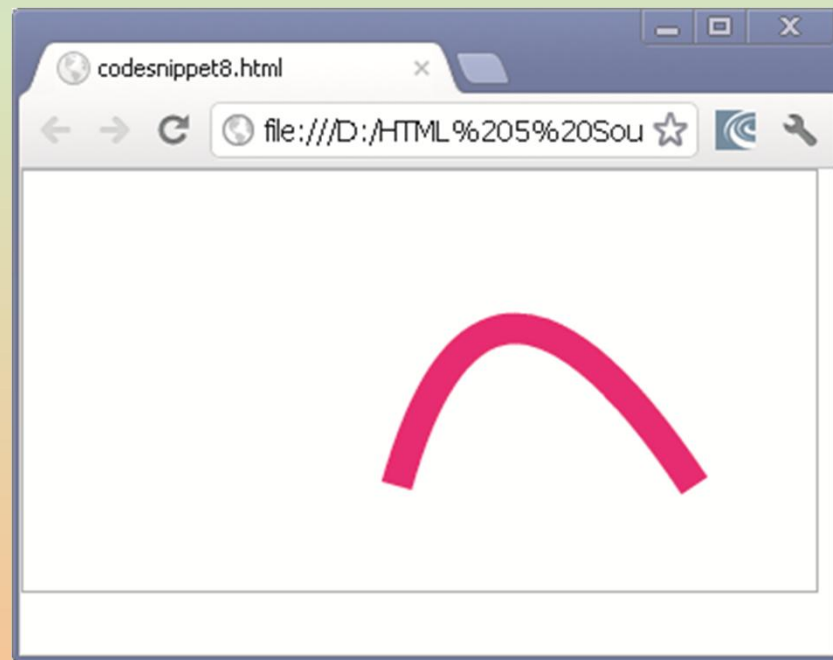
```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body
      {
        margin: 0px;
        padding: 0px;
      }
      #mCanvas
      {
        border: 1px solid #9C9898;
      }
    </style>
  </head>
  <body>
    <div>
      <canvas id="mCanvas">
        <!-- Drawing a quadratic curve -->
      </canvas>
    </div>
  </body>
</html>
```

# Working with Drawing Objects 16-17

```
    window.onload = function() {  
        var canvas = document.getElementById("mCanvas");  
        var ctext = canvas.getContext("2d");  
        ctext.beginPath();  
        ctext.moveTo(178, 150);  
        ctext.quadraticCurveTo(220, 0, 320, 150);  
        ctext.lineWidth = 15;  
        // line color  
        ctext.strokeStyle = "Fuchsia";  
        ctext.stroke();  
    };  
</script>  
</head>  
<body>  
    <canvas id="mCanvas" width="378" height="200"></canvas>  
</body>  
</html>
```

# Working with Drawing Objects 17-17

- In the code, the control point defines the curve of the quadratic by two tangential lines that are connected to both the context point and the end point.
- The context point is represented using the `moveTo()` method.
- This method moves the control point from the context point and the end point to create a sharper curve.
- It also moves the control point close to the context point and end point to create broad curves.
- Following figure displays a quadratic curve in a canvas.



# HTML5 Working with Images 1-3

- In HTML5, the user can draw image objects on canvas using the `drawImage ()` method.
- The `drawImage ()` method can also draw parts of an image and increase or reduce the size of the image.
- This method accepts nine parameters, depending on editing that is required on the image.
- The image object can be a video, an image, or another canvas element.
- The Code Snippet demonstrates how to create an image using HTML5.

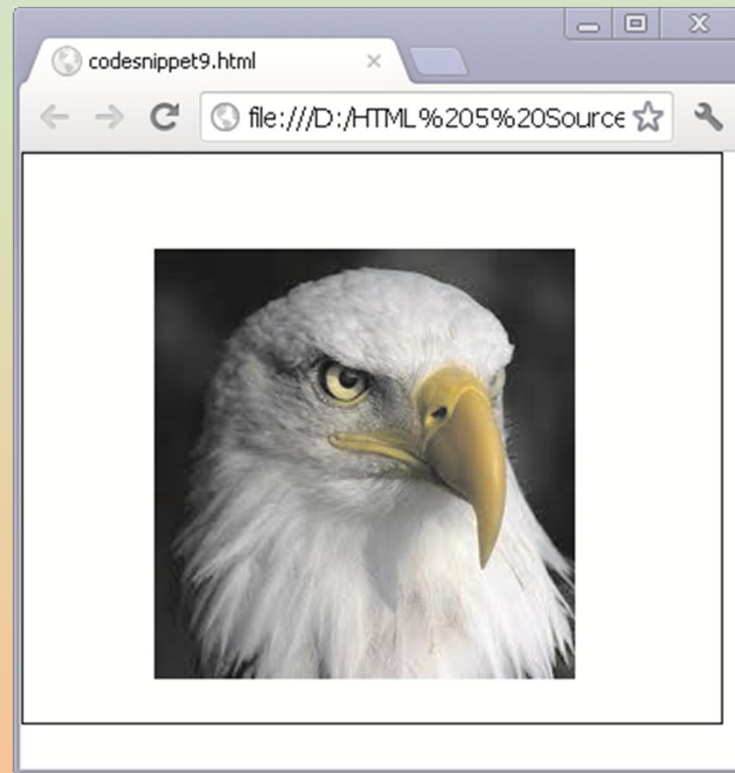
```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body {
        margin: 0px;
        padding: 0px;
      }
      #mCanvas {
        border: 1px solid #9C9898;
      }
    </style>
```

# HTML5 Working with Images 2-3

```
<script>
    window.onload = function()
    {
        var canvas = document.getElementById("mCanvas");
        var ctext = canvas.getContext("2d");
        var imgObj = new Image();
        imgObj.onload = function()
        {
            ctext.drawImage(imgObj, 69, 50);
        };
        imgObj.src = "bird.jpg";
    };
</script>
</head>
<body>
    <canvas id="mCanvas" width="368" height="300"></canvas>
</body>
</html>
```

# HTML5 Working with Images 3-3

- In the code, the `onload` property is used.
- The source of the object is defined by using the `src` property.
- The image has to be loaded first and then instantiated with the `drawImage()` method.
- This method takes image object as the parameter with the x and y coordinates of the image.
- Following figure displays an image drawn on a HTML5 canvas.





# HTML5 Working with Text 1-5

- HTML5 canvas enables you to set the font, style, and size of text by using the font properties.
- The font style can be italic, normal, or bold.
- To set the text color, the fillStyle property of the canvas can be used.
- The Code Snippet demonstrates how to set the font, size, style, and color of the text on a HTML5 canvas.

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body {
        margin: 0px;
        padding: 0px;
      }
      #mCanvas {
        border: 1px solid blue;
      }
    </style>
```

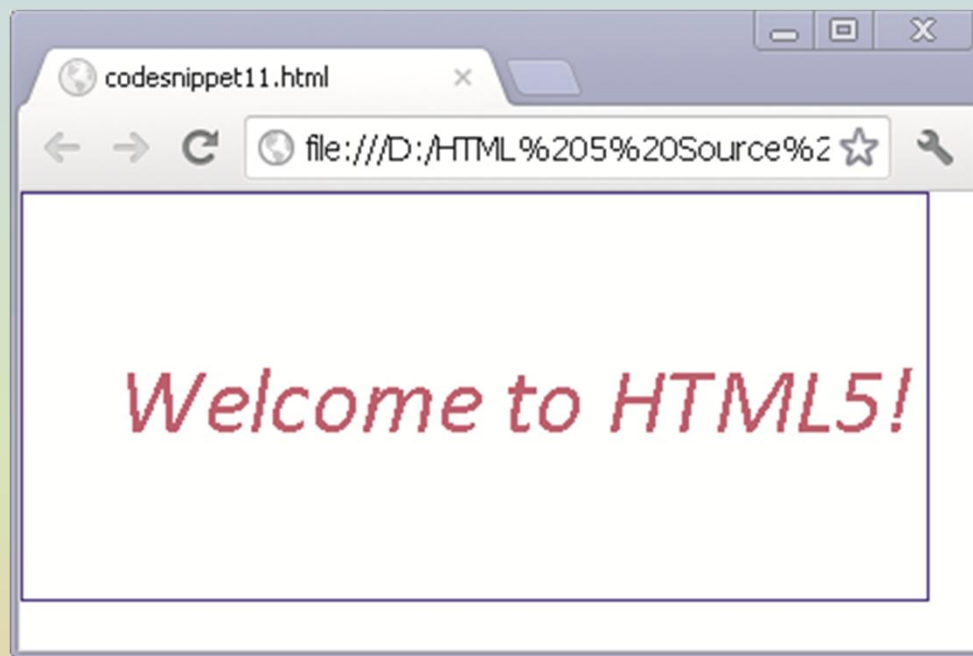
# HTML5 Working with Text 2-5

```
<script>
    window.onload = function() {
        var canvas = document.getElementById("mCanvas");
        var ctext = canvas.getContext("2d");
        ctext.font = "italic 30pt Calibri";
        ctext.fillStyle = "MediumVioletRed";
        ctext.fillText("Welcome to HTML5!", 40, 100);
    };
</script>
</head>
<body>
    <canvas id="mCanvas" width="380" height="170"></canvas>
</body>
</html>
```

- In this code, the font text is specified as `Calibri`, style as `italic`, and size is set to `30pt`.
- The `fillStyle` property specifies the text color and the `fillText` property is used to set the text on the canvas.

# HTML5 Working with Text 3-5

- Following figure displays working with text in a HTML5 canvas.



- In HTML5 canvas, the user can set the stroke color by using the `strokeText()` method and `strokeStyle` property of the canvas context.

# HTML5 Working with Text 4-5

- The Code Snippet demonstrates the use of stroke text in HTML5 canvas.

```
<!DOCTYPE HTML5>
<html>
  <head>
    <script>
      var y = 110;
      ctext.font = "40pt Calibri";
      <style>
        ctext.lineWidth = 2;
        /body stroke color
        ctext.strokeStyle = "Brown";
        ctext.strokeText("HTML5", x, y);
      </script>
    </head>
    <body>
      <canvas id="mCanvas" width="360" height="200"></canvas>
    </body>
  </html>

  window.onload = function() {
    var canvas = document.getElementById("mCanvas");
    var ctext = canvas.getContext("2d");
```

# HTML5 Working with Text 5-5

- In this code, the stroke color is set by using the `strokeStyle` property and the `strokeText()` method.
- Following figure displays the stroke text in HTML5 canvas.



# Using Transparency for Text in Canvas 1-3

- There are two ways to set the transparency for the text and shapes.
- The first method is to use the `strokeStyle` and `fillStyle` by using the `rgb` function.
- The second method is to use `globalAlpha` drawing state property, which can be applied universally.
- The `globalAlpha` property is a value that ranges between 0 (fully transparent) and 1 (fully opaque).
- The Code Snippet demonstrates the use of `globalAlpha` property.

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body {
        margin: 0px;
        padding: 0px;
      }
      #mCanvas {
        border: 1px solid black;
      }
    </style>
```

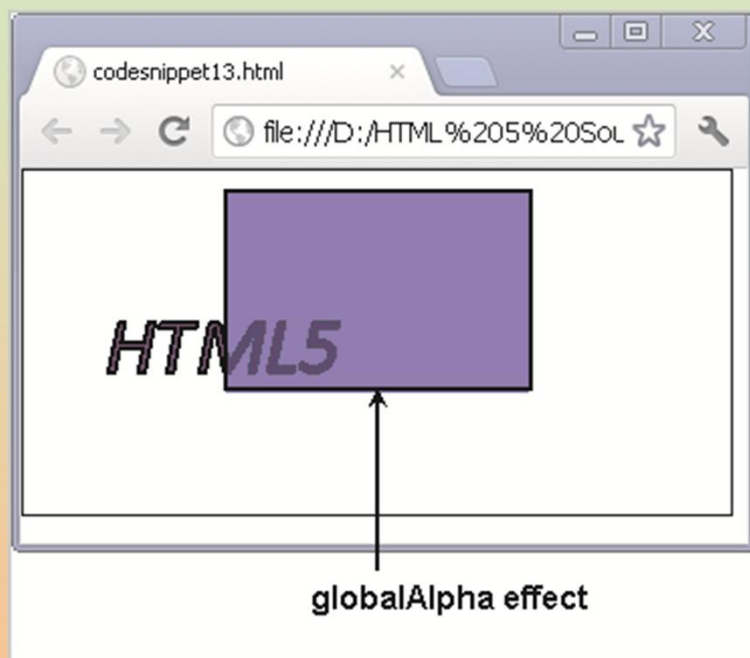


# Using Transparency for Text in Canvas 2-3

```
<script>
    window.onload = function()
    {
        var canvas = document.getElementById("mCanvas");
        var ctext = canvas.getContext("2d");
        ctext.fillStyle = "Indigo";
        ctext.strokeStyle = "black";
        ctext.lineWidth=2;
        ctext.font = "italic 30pt Calibri";
        ctext.fillText("HTML5", 40, 100);
        ctext.strokeText("HTML5", 40, 100);
        ctext.fillStyle="blue";
        ctext.globalAlpha=0.5;
        ctext.fillRect(100, 10, 150, 100);
    };
</script>
</head>
<body>
    <canvas id="mCanvas" width="350" height="170"></canvas>
</body>
</html>
```

# Using Transparency for Text in Canvas 3-3

- In the code, the `fillStyle` and `strokeStyle` is used to color the text.
- The 'HTML5' text `lineWidth` is specified as 2 and the `font-family` is set to Calibri with `italic` style and `font-size` to 30pt.
- The `fillText` property fills the color and `strokeText` property applies the stroke color to the HTML5 text.
- The `fillStyle` is set to blue and `globalAlpha` property is set to 0.5.
- The `fillRect(100, 10, 150, 100)` specifies the x, y, height, and width of the rectangle.
- Following figure displays the stroke text in HTML5 canvas.





# Using Events with jQuery 1-6

- jQuery also offers different events to deal with common interactions when the user moves the mouse or switches between two actions while clicking.
- The following are the events:

## ➤ **hover() event**

- The mouseenter and mouseleave are the two events often used together.
- jQuery provides a `hover()` function that accepts two parameters.
- The first parameter executes when the mouse moves over the element and the second function executes when the mouse moves away from the element.
- The Code Snippet demonstrates the hover event.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="jquery-1.7.2.min.js"></script>
    <script>

      $(document).ready(function() {
```

# Using Events with jQuery 2-6

```

$ ("p").hover(function() {
    $ ("p").css("background-color","red");
},function() {
    $ ("p").css("background-color","maroon");
});
</script>
</head>
<body>
    <p>Hover the mouse on this line.</p>
</body>

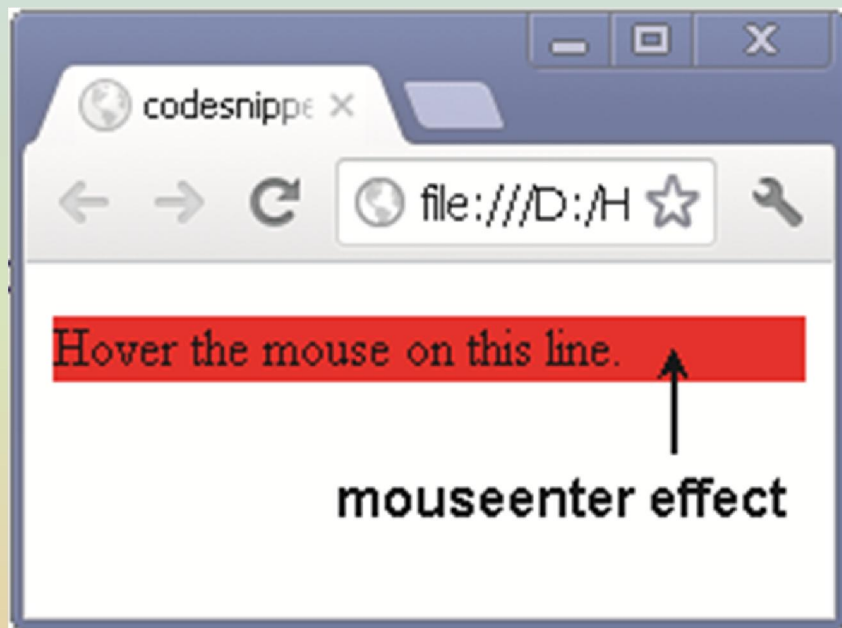
</html>

```

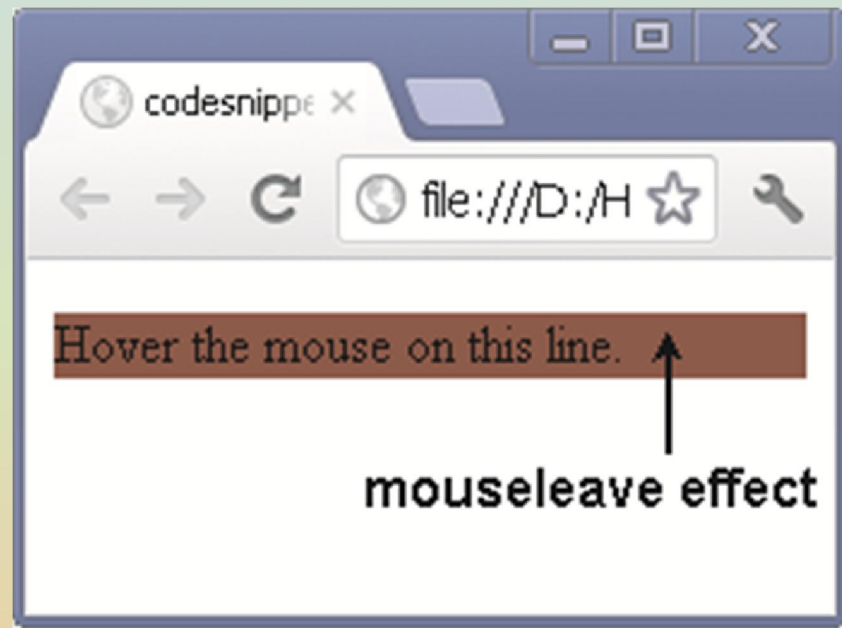
- In the code, the `hover()` method is used.
- When the mouse is placed on the text, then the background color changes to red.
- When the user moves the mouse outside the text, the background color changes to maroon.

# Using Events with jQuery 3-6

- Following figure displays the mouseenter effect.



- Following figure displays the mouseleave effect.





# Using Events with jQuery 4-6

## ➤ toggle() event

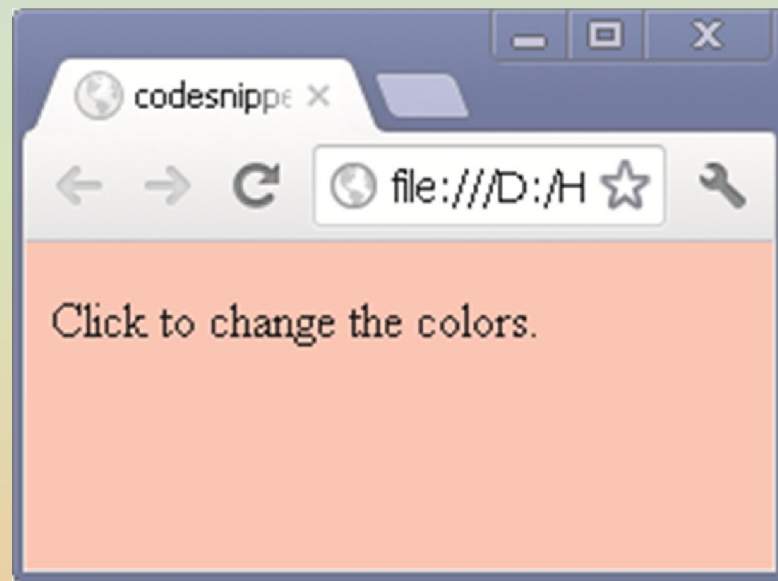
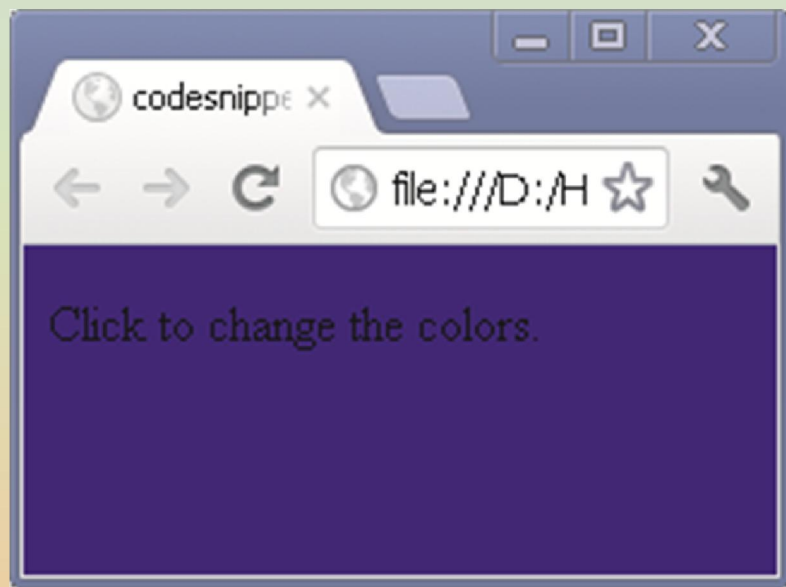
- The `toggle()` event works in a similar manner as that of the `hover()` event, except that it responds to mouse clicks.
- The `toggle()` function accepts more than two functions as arguments.
- All the functions passed to the `toggle()` event will react to its corresponding click action.
- The Code Snippet demonstrates the toggle event.

```
<!DOCTYPE html>
<html>
<head>
  <script src="jquery-1.7.2.min.js"></script>
</head>
<body>
  <p>Click to change background color</p>
  <button>Click</button>
  </body>
</html>

<script>
  $(document).ready(function() {
    $("p").toggle(function() {
      $("body").css("background-color", "blue");
    }, function() {
      $("body").css("background-color", "pink");
    });
  });
</script>
```

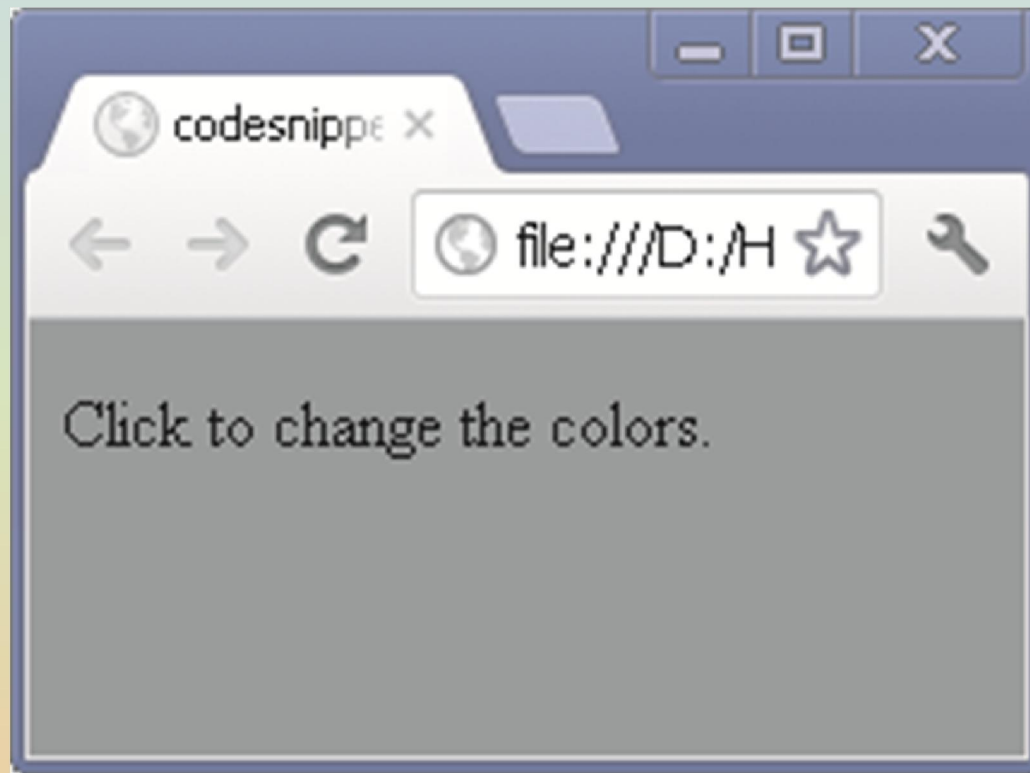
# Using Events with jQuery 5-6

- In the code, the `toggle()` method is used.
- When the user clicks the text then the `background-color` of the document is changed to blue, pink, and grey respectively.
- Following figure displays the toggle effect to blue.
- Following figure displays the toggle effect to pink.



# Using Events with jQuery 6-6

- Following figure displays the toggle effect to grey.



# Inclusion of External Content in Web Pages

HTML5 introduces the `<eventsource>` tag that allows the user to push external content in the Web page. This model is referred to as push model.

Since the `<eventsource>` tag is not supported in many browsers, users make use of the `<embed>` tag for this purpose.

The `<embed>` tag is a new element in HTML5 and it is represented as a container for an interactive content or an external application.

The `<embed>` tag is often used to add elements such as image, audio, or video on a Web page.

- The Code Snippet demonstrates the use of `<embed>` tag.

```
<embed src="mymovie.mp3" />
```

- In this code, the `src` attribute specifies the path of an external file to embed.

- The `<canvas>` element is a drawing area where the user can draw graphics, use images, add animations, and also add text for enhancing the user experience on Web pages.
- To create a line, on a canvas one can use the `stroke()`, `beginPath()`, `lineTo()`, and `moveTo()` methods.
- Arcs are represented using a start angle, an end angle, a radius, a center point, and the drawing direction (anticlockwise or clockwise).
- With HTML5 canvas, the user can create a rectangle using the `rect()` method.
- Bezier curves are represented with the two control points, context points, and an end point.
- HTML5 canvas allows the user to create quadratic curves using the `quadraticCurveTo()` method.
- HTML5 canvas enables the user to draw image object on canvas using the `drawImage()` method.