Session: 13

Canvas and Web Storage in HTML5

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
- Explain Web storage in HTML5
- Explain session storage
- Explain local storage
- Explain the Indexed DB API
- Describe a native app
- Explain the difference between native apps and HTML5 apps
- Describe the advantages of native and HTML5 apps
- List the steps to convert HTML5 apps to native apps

Canvas Element 1-3

The `<canvas>` element in HTML5 can be used to draw shapes on Websites 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 `<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, drawing is done using a JavaScript code.

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

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

Canvas Element 2-3

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

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 3-3

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');
        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>
```

Drawing a Line in Canvas 1-3

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

Syntax:

```
ctext.beginPath();
ctext.moveTo(x,y);
ctext.lineTo(x,y);
ctext.stroke();
```

where,

- `ctext` - 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 2-3

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

```
<!DOCTYPE HTML>
<html>
  <head>
  <title>Line</title>
    <style>
      body
      {
        margin: 0px;
        padding: 0px;
      }
      #mCanvas
      {
        border: 1px solid red;
      }
    </style>
    <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>
  </head>
  <body>
    <canvas id="mCanvas" width="360" height="200"></canvas>
  </body>
</html>
```

Drawing a Line in Canvas 3-3

- 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.

Working with Drawing Objects 1-11

- 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
`fillStyle`	The values for this property can be gradient, pattern, or a CSS color. The default property style is solid black, but user can set color according to the requirements.
`fillRect(x, y, width, height)`	Enables the user to draw a rectangle with the existing fill style.
`strokeStyle`	The values for this property can be gradient, pattern, or a CSS color.
`strokeRect(x, y, width, height)`	Enables the user to draw a rectangle with the existing stroke style. This property is used to draw the edges of the rectangle.
`clearRect(x, y, width, height)`	Used to clear the pixels in a rectangle.

Working with Drawing Objects 2-11

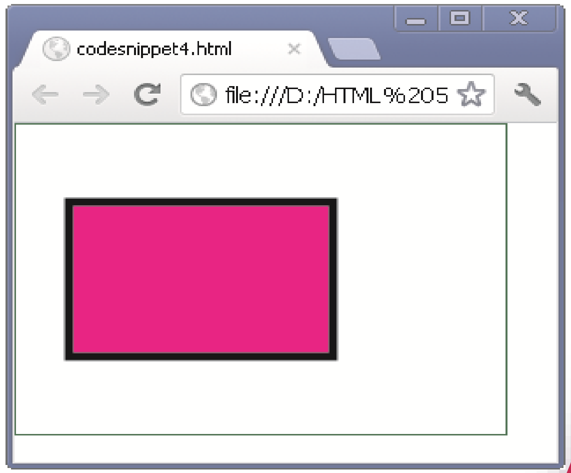
- The Code Snippet demonstrates how to create a rectangle in HTML5 canvas.

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      #mCanvas {
        border: 1px solid green;
      }
      body {
        margin: 0px;
        padding: 0px;
      }
    </style>
   <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 3-11

- 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.

Output:



Working with Drawing Objects 4-11

➢ 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

Working with Drawing Objects 5-11

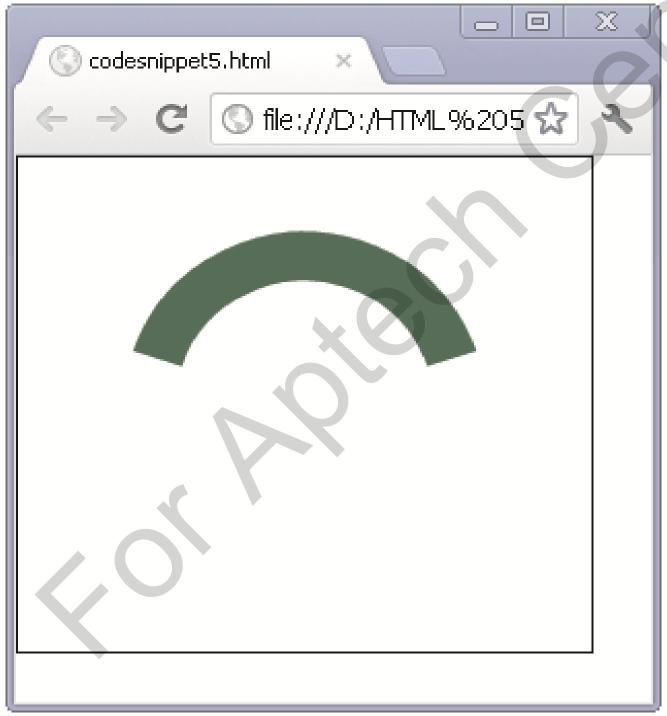
- The Code Snippet demonstrates how to create an arc in HTML5 canvas.

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body
      {
        margin: 0px;
        padding: 0px;
      }
      #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 6-11

- 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.



Working with Drawing Objects 7-11

➢ 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 * 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 8-11

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

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      body {
        margin: 0px;
        padding: 0px;
      }
      #mCanvas
      {
        border: 1px solid blue;
      }
    </style>
    <script>
      window.onload = function() {
        var canvas = document.getElementById("mCanvas");
        var ctext = canvas.getContext("2d");
        var ctrX = canvas.width / 2;
       var ctrY = canvas.height / 2;
        var radius = 70;
        ctext.beginPath();
        ctext.arc(ctrX, ctrY, radius, 0, 2 * Math.PI, false);
        ctext.fillStyle = "DarkOrchid";
        ctext.fill();
        ctext.lineWidth = 4;
        ctext.strokeStyle = "black";
        ctext.stroke();
     };
    </script>
  </head>
  <body>
    <canvas id="mCanvas" width="356" height="150"></canvas>
  </body>
</html>
```

Working with Drawing Objects 9-11

- 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*PI`.
- The `anticlockwise` defines the direction of the path of an arc between the two start and end points.

➢ 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.

Working with Drawing Objects 10-11

- 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 11-11

➢ 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;
      }
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 Images

- 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.
- The image object can be a video, an image, or another canvas element.

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

- The Code Snippet demonstrates how to create an image using HTML5.

Working with Text 1-2

- 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>
<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>
```

Working with Text 2-2

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

```
<!DOCTYPE HTML>
 <html>
  <head>
    <style>
      body {
        margin: 0px;
        padding: 0px;
      }
      #mCanvas {
        border: 1px solid black;
      }
    </style>
    <script>
        window.onload = function() {
        var canvas = document.getElementById("mCanvas");
        var ctext = canvas.getContext("2d");
        var x = 80;
        var y = 110;
        ctext.font = "40pt Calibri";
        ctext.lineWidth = 2;
        // stroke color
        ctext.strokeStyle = "Brown";
        ctext.strokeText("HTML5", x, y);
      };
    </script>
  </head>
  <body>
    <canvas id="mCanvas" width="360" height="200"></canvas>
  </body>
</html>
```

Using Transparency for Text in Canvas

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>
<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 Events with jQuery 1-3

- jQuery also offers different events to deal with common interactions when the user moves the mouse or switches between two actions while clicking.
- 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(){
$("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>
```

Using Events with jQuery 2-3

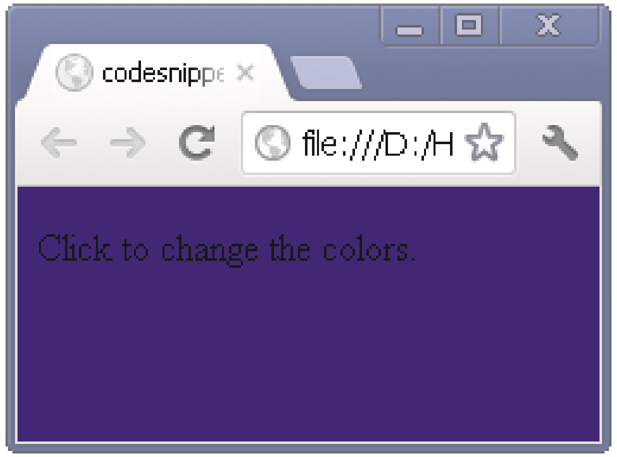
➢ toggle() event

- 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>
   <script>
    $(document).ready(function(){
    $("p").toggle(function(){
    $("body").css("background-color","blue");},
    function(){
    $("body").css("background-color","pink");},
     function(){
    $("body").css("background-color","grey");}
    );
   });
   </script>
  </head>
   <body>
    <p>Click to change the colors.</p>
   </body>
 </html>
```

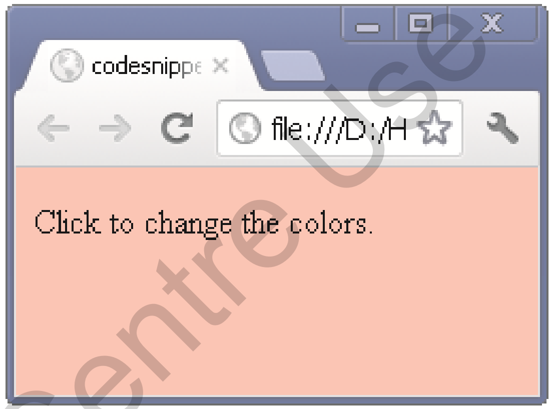
Using Events with jQuery 3-3

- Following figure displays the toggle effect to blue:



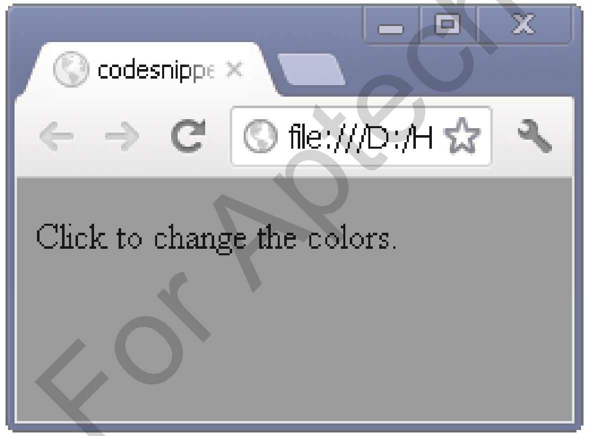
Click to change the colors.

- Following figure displays the toggle effect to pink:



Click to change the colors.

- Following figure displays the toggle effect to grey:



Click to change the colors.

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.

Cookies and Web Storage

Traditionally, over the last few decades, Web applications have been using cookies to store small amounts of information on a user's computer.

A cookie is a file that stores user-related information and may either be temporary or permanent.

A cookie can be created for login details which can be saved for a specified period on a user's computer.

- Drawbacks of cookies are as follows:
  - Cookies slow down the performance of Web application, as they are included with every HTTP request
  - Cookies cannot be considered as safe means for transmission of sensitive data
  - Cookies cannot store large amount of information, as they have a limitation of size of four kb
- W3C has designed a specification named Web Storage API which offer a solution to store data on the client-side

Is a W3C specification and certain browsers refer to it as 'DOM Storage'.

Provides functionality for storage of data on the client-side that is on user's machine.

Stores data that can cater for both temporary as well as permanent needs.

Offers more control than traditional cookies, and is easy to work with.

Was originally a part of the HTML5 specification, but now it is present in a separate specification and stores a maximum of five mb of information per domain.

Web Storage versus Cookies

- Some key differences between cookies and Web storage are as follows:

Cookies are meant to be read on the server-side, whereas Web storage is available only on the client-side.

Cookies are sent along with each HTTP request to the server, whereas Web storage data is not carried over to the server.

Cookies result in bandwidth overhead and thus lead to high costs, as they are sent with each HTTP request. The Web storage is stored on the user's hard drive, so it costs nothing to use.

With cookies, the information data that could be stored is four kb, whereas with Web storage, a large amount of data can be stored upto five mb.

Browser-specific Web Storage

Web storage is browser-specific and the location where the Web storage data is stored depends on the browser.

Each browser's storage is separate and independent, even if it is present on the same machine.

HTML5 Web storage is implemented natively in most Web browsers, so one can use it even when third-party browser plug-in is not available.

- Following table lists the support of various browsers for HTML5 Web storage:

Browser	Version
IE	8.0+
Firefox	3.6+
Safari	4.0+
Chrome	5.0+
Opera	10.5+

Exploring Web Storage

Two types of HTML5 Web storage are namely, session storage and local storage.

Both session and local storage enable to store around five mb of data per domain.

To check for browser support of HTML5 Web storage, a property named **localStorage** or **sessionStorage** is available as a global variable for the window object.

If there is no support, the **localStorage** or **sessionStorage** property will be undefined.

Code Snippet demonstrates the script to check the support for HTML5 Web storage in the browser.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Support for Web Storage</title>
    <script>
    function checkSupport() {
      if (('sessionStorage' in window) && window['sessionStorage']
   !== null)
      {
        alert("Your browser supports Web Storage");
        return;
      }
      alert("Your browser does not support Web Storage");
    }
    </script>
  </head>
  <body onload="checkSupport();">
  </body>
</html>
```

Session Storage 1-3

Keeps track of data specific to one window or tab and discards it as soon the user closes the tab (or window) that he/she was working with.

Lasts for the entire duration of the session and hence, is not persistent.

Makes use of named key/value pairs which are enclosed within double quotes.

Stores the data using the named key, whereas the data is retrieved by referring to that key.

Key is a string, whereas the value stored in the key can be of any data type such as string, boolean, integer, or float. Regardless of the type of data that is stored, it is actually stored internally as a string.

Storing and retrieving data of other types requires the use of functions to convert them into the appropriate data types.

- Following table lists some examples of named key/value pairs belonging to various data types:

Key	Value
Name	Sarah
book	C Programming
Email	info@me.com
car	Toyota Innova
age	28
uservalid	true

Session Storage 2-3

Storing and retrieving data - `setItem()` and `getItem()` methods are used to store and retrieve data from session storage respectively.

- Syntax to use `setItem()` and `getItem()` methods is as follows:

- **To assign data**

```
sessionStorage.setItem(key, value);
```

where,

`key`: Is the named key to refer to the data.

`value`: Is the data to be stored.

- **To retrieve data**

```
var item = sessionStorage.getItem(key);
```

where,

`item`: Is the variable into which the data will be saved.

`key`: Is the named key to refer to the data.

- **To remove data**

```
sessionStorage.removeItem(key);
```

where,

`key`: Is the named key to refer to the data.

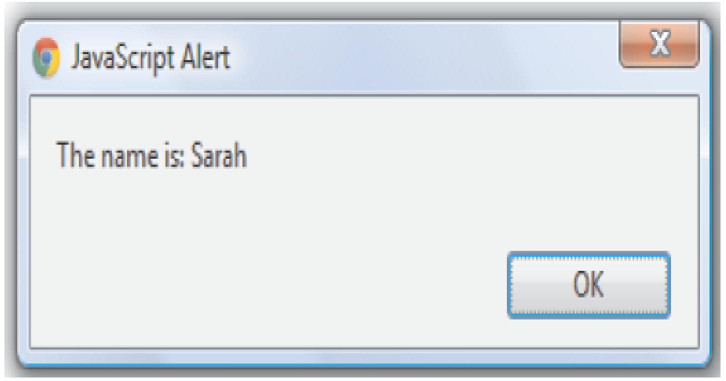
- **To clear data**

```
sessionStorage.clear();
```

Session Storage 3-3

- **Code Snippet demonstrates how to set and retrieve a name using** `sessionStorage` **object.**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Working with Session Storage</title>
  <script>
  function testStorage() {
    if (('sessionStorage' in window) &&
   window['sessionStorage'] !== null)
    {
      sessionStorage.setItem('name', 'Sarah');
      alert('The name is: ' +
   sessionStorage.getItem('name'));
    }
  }
  </script>
  </head>
  <body onload=" testStorage();">
  </body>
</html>
```



The name is: Sarah

Local Storage 1-3

Enables to save data for longer periods on the user's computer, through the browser.

Data is persistent and can be retrieved when a user visits the site again.

Is used, if data needs to be stored for more than a single session.

Works in a similar fashion as session storage and uses similar functions, such as `setItem()`, `getItem()`, `removeItem()`, and `clear()`.

Local Storage 2-3

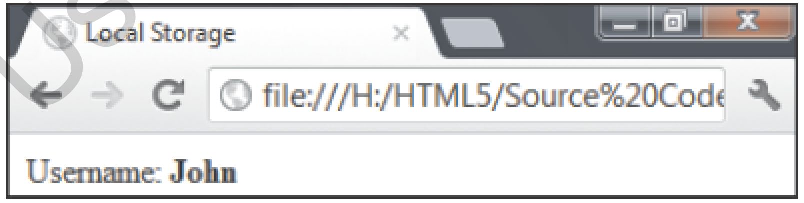
- Code Snippet demonstrates use of local storage to store the value of **username** field and later, retrieve the value in another Web page.

```
<!DOCTYPE html>
<html>
  <title> Local Storage </title>
  <script>
  function store() {
      if  (('localStorage'  in  window)  &&  window['localStorage']  !==
   null) {
        var username = document.getElementById('username').value;
        localStorage.setItem('username', username);
      } else {
        alert ('your browser does not support storage');
      }
    }
    function cancel_store() {
      if  (('localStorage'  in  window)  &&  window['localStorage']  !==
   null) {
        localStorage.removeItem('username');
      } else {
        alert ('your browser does not support storage');
      }
    }
</script>
  </head>
   <body>
    <form method="get" action="success.html">
      Username: <input type="text" id="username" value="" size="20"
  onblur="store()"/>
      <input type="submit" value="Submit"/>
      <input type="reset" Value="Cancel" onclick="cancel_store()"/>
  </body>
</html>
```

Local Storage 3-3

- Code Snippet shows the success.html page that retrieves value from the local storage and displays it in the browser.

```
<!DOCTYPE html>
<head>
  <title> Local Storage </title>
  <script>
    function print() {
      var username = localStorage.getItem('username');
      document.getElementById('lblMsg').innerHTML = 'Username:
       is <b>'+ username+'</b>';
    }
  </script>
</head>
<body onload="print()">
  <label id="lblMsg"></label><br>
</body>
</html>
```



Username: **John**

Indexed Database API 1-3

A database is an organized collection of data.

Databases, such as relational database stores the data in the form of tables.

A table comprises rows and columns that are used to store data.

The representation of data from a table is in the form of records.

HTML5 has introduced a new Web Storage API which can host Web databases locally within the user browser.

Web databases are not like relational databases in terms of functionality.

Indexed Database API 2-3

Indexed Database API is a specification also known as IndexedDB.

It is basically an object store that can be used to store and manipulate data on the client-side.

The object store is the primary storage mechanism that stores the object in the database managed locally within the browser.

It enables to create an object store of a particular type in which objects can be persisted using JavaScript.

IndexedDB enables to create Web applications with rich query abilities and which can work both online and offline.

IndexedDB supports two types of API namely, synchronous and asynchronous.

The synchronous API can be used with WebWorkers, whereas asynchronous API can be used for Web applications.

Indexed Database API 3-3

IndexedDB API is implemented using **window.indexedDB** object.

Browsers implement the IndexedDB object with their own prefixes. For example, Chrome browser uses the **webkit** prefix, whereas Mozilla supports **–moz** prefix.

- Following table lists the browser support for the IndexedDB API:

IE	Firefox	Chrome	Safari	Opera	iOS Safari
6.0	-	-	-	-	3.2
7.0	8.0moz	-	-	-	4.0-4.1
8.0	9.0moz	16.0webkit	5.0	-	4.2-4.3
9.0	10.0moz	17.0webkit	5.1	11.6	5.0
10.0	11.0moz	18.0webkit	6.0	12.0	-
-	12.0moz	19.0webkit	-	-	-

Indexed DB API 1-2

- Some of the basic constructs of IndexedDB API are as follows:

Database	The IndexedDB database comprises more than one object store. Each database contains a name that identifies the origin of the database and a version number which identifies the lifetime of the database.
Object Store	Is the main mechanism to store data in a database. They hold the data stored in the database in the form of records.
Keys	Each record stored in the database is identified by a unique key.
Values	Are the data stored in the records.
Key Path	Is a string that defines how the browser should extract key from a value. The key from a value can be extracted either in the object store or index.
Index	Is used when the data from the object store is retrieved based on some other values other than a key.
Transaction	Any addition or retrieval of the data in a database is performed by using transaction. Each transaction has a mode, scope, and a request list.

Indexed DB API 2-2

- Some of the basic constructs of IndexedDB API are as follows:

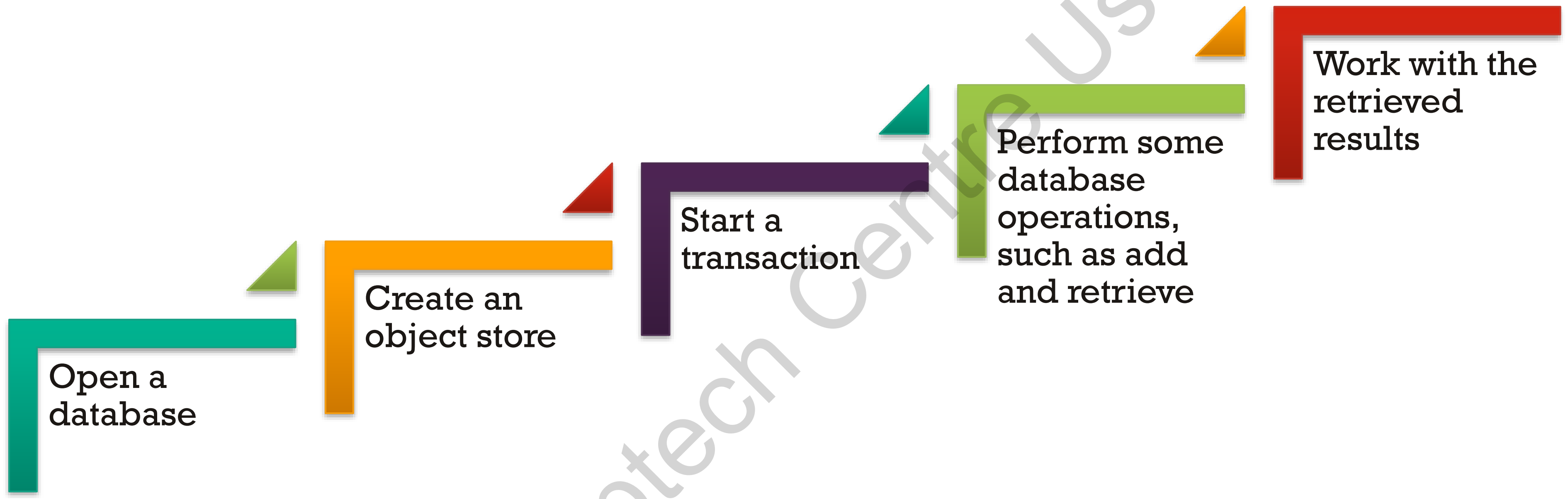
Requests - Operations, such as reading or writing on the database is performed using a request. Each request contain attributes, such as flag, source object, result, and error.

Cursor - Is a mechanism used to retrieve multiple records from a database.

Key Range - Records from the object stores and indexes are retrieved using keys or key ranges. A key range refers to retrieval of data between specified bounds based on the keys.

Implementing IndexedDB API 1-5

- Steps to implement the IndexedDB API in a Web application are as follows:


Open a database
Create an object store
Start a transaction
Perform some database operations, such as add and retrieve
Work with the retrieved results

Implementing IndexedDB API 2-5

- **Opening a Database**
- Code Snippet shows the code to open a database

```
var   indexedDB   =   window.indexedDB   ||   window.webkitIndexedDB   ||   window.mozIndexedDB   ||
window.msIndexedDB;
var request = indexedDB.open("CompanyDB", 1);
request.onsuccess = function (event) {
. . .
};
request.onerror = function (event) {
console.log("IndexedDB error: " + event.target.errorCode);
};
```

- **Updating Version of a Database**

After the database is opened, it can be structured by providing a version number which helps to set up the database.

- Code Snippet shows the code that specifies the version number to the database

```
var setVrequest = db.setVersion("1.99");
setVrequest.onsuccess = function(event) {
. . .
}
```

Implementing IndexedDB API 3-5

- **Creating the Object Store**

Structure of IndexedDB database facilitates the storage of multiple object stores.

Object store is created using createObjectStore() method which accepts two arguments namely, the store name and a parameter object.

- Code snippet demonstrates the code to create an object store named employee in the CompanyDB database.

```
var employeeData = [
{ name: "John Smith", email: "john@company.com" },
{ name: "Jill Patrick", email: "jill@company.com" },
{ name: "Rock Ethan", email: "rock@company.com" },
{ name: "Daniel Andrew", email: "daniel@company.com" }
];
var objectStore = db.createObjectStore("employee", {
keyPath: "id", autoIncrement: true });
for (i in employeeData) {
  objectStore.put(employeeData[i]);
  alert("Record added");
}
```

Implementing IndexedDB API 4-5

- **Creating a Transaction**

To perform database operation, such as retrieving data from the object store, IndexedDB provides a IDBTransaction object.

This object can be created in three mode namely, read-only, read-write, and snapshot.

- **Code Snippet demonstrates the code to retrieve data from the employee object store using `get()` function of the transaction object.**

```
var trans = db.transaction(["employee"],
IDBTransaction.READ_WRITE).objectStore("employee");
var request = trans.get(2);
request.onerror = function(event) {
  // Handle errors!
};
request.onsuccess = function(event) {
  // Do something with the request.result!
  alert("Name: " + request.result.name);
};
```

Implementing IndexedDB API 5-5

- **Opening a Cursor**

Cursor is used to retrieve multiple records from an object store.

They can be used when the value of key path is not known. They are part of a transaction and are opened for a particular object store.

- Code Snippet demonstrates the code to retrieve multiple records from the employee object store.

```
store = db.transaction("employee").objectStore("employee");
store.openCursor().onsuccess = function(event) {
  var cursor = event.target.result;
  if (cursor) {
    alert("Name for id " + cursor.key + " is " + cursor.value.name);
    cursor.continue();
  }
};
```

Limitations of IndexedDB API

- Design limitations for IndexedDB API used for client-side storage of data are as follows:

Internationalized sorting deals with sorting of string data. As the database does not follow any international order for storing data, internationalized sorting is not supported by the API.

IndexedDB API does not synchronize client-side database with the server-side databases.

IndexedDB API supports querying the client-side database, but does not support the use of operators, such as LIKE that is used by Structured Query Language (SQL).

Converting HTML5 Apps to Native Apps

A native application also known as native app is an application program that is built for using it on a particular device or platform.

A native app, when compared with Web app is installed on a device and has a faster response, because it has a direct user interface.

Difference between Native Apps and HTML5 Apps

HTML5 Web apps are accessible and used on any devices through Web browser similar to the mobile Website.

Web apps have the ability of offline access which means that the user need not have a network connection.

- Following table lists differences between native apps and HTML5 apps:

Native Apps	HTML5 Apps
Native Apps runs on iOS and Android devices that can be downloaded or purchased from the online app stores.	HTML5 Apps runs on a Web server, usually in a Web browser.
Native Apps use programming language, such as Java for Android devices and Objective C for iOS devices.	Web developers use HTML, JavaScript, and CSS. They need to acquire the skills of Java and objective C for writing native applications.

Advantages of HTML5 Apps

- Some of the reasons to develop HTML5 applications are as follows:

Users cannot identify the differences	Cannot identify whether they are working on a hybrid HTML5-native application or a fully native application or an HTML5 application.
Users adjust styles for devices	HTML5 apps can be viewed on any devices that contains Web browser.
Upcoming functionalities	HTML5 does not support all features on a device, but it is coming up with new functionalities.
Improving Performance	Many developers learn new methods to improve the performance of Web.
Independent device	If the developers want that their application to be used by a large number of users, then they should design and develop applications for both mobile users as well as desktop users.
Developers are not locked in app stores	HTML5 developers are not restricted to an app store. Instead, they can create applications and sell them like any other Web page.

Advantages of Native Apps

- Major advantage of native apps over HTML5 apps is that they are faster than HTML5 apps. Native apps provide more benefits over HTML5 apps. These are as follows:

Providing access to device hardware	Uploading Files	Push notifications	Accessing device files	Superior graphics than HTML5	Offline access
There are no APIs available for accelerometers, cameras, or any other device hardware for HTML5 apps.	Native apps can access the file system in Android and some files in iOS. However, the HTML5 file API does not work on Android or iOS.	The push notifications are sent always with an open IP connection to applications on the iOS device.	Native apps communicate with files on devices, such as contacts and photos. However, these files cannot be seen from HTML5 app.	HTML5 has a canvas element, but it will not create a full 3D experience.	HTML5 provides access to offline Web applications. However, a native app is stored on local machine, so that users do not require access to the Web to work with the application.

Converting HTML5 Apps to Native Apps

- Users have a choice of developing their application in HTML5 and convert them into a native app
- Users can use some tools to convert an HTML5 app to Native app and they are as follows:

PhoneGap

Is an HTML5 app that allows the user to create native apps with Web technologies and is accessible to app stores and APIs.

Appcelerator

Is a cross-platform mobile application development support and allows the users to create Android, iOS, and mobile Web apps.

Summary

- 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.
- Web Storage is a W3C specification that provides functionality for storing data on the client-side for both temporary as well as permanent needs.
- HTML5 Web applications make use of Web storage to implement client-side persistent storage and they are: session storage and local storage.
- Session storage keeps track of data specific to one window or tab.
- The setItem() and getItem() methods are used to store and retrieve the data from session storage.
- Local storage enables to save data for longer periods on the user's computer, through the browser.
- IndexedDB API is basically an object store that can be used to store and manipulate data on the client-side.
- A native application also called as native app is an application program that is built for a particular device or platform.