K Srinivas
Vasu34k@gmail.com

HTML Course Contents

1. HTML5 Overview
2. Scoping the range of HTML5 features
3. An ideal future: common HTML everywhere
4. Current and imminently-available support for HTML
5. New Elements in HTML5
6. Introducing the new tags
7. Approaches to writing HTML5 code
8. The Canvas
9. Positioning content with the Canvas
10. Drawing with Javascript
11. Understanding Co-ordinates
12. Controlling layout, visibility and state with Canvas tags
13. Storing Data on the Client
14. Persistent client-side data storage
15. Session management
16. Security Issues
17. Updated Tags in HMTL5
18. Tags and attributes which still work as before
19. Modified behaviour of some tags
20. Forms
21. Input Types: Email, URL, Number, Range and Date fields
22. Datalist, Keygen fields
23. Other Form Elements and Attributes: AutoComplete and Form Override
24. Media
25. Web worker
26. App Cache
27. Audio and Video
28. GeoLocation
29. Drag and Drop
30. Server-sent Events

K Srinivas
Vasu34k@gmail.com

Nuts and Bolts of HTML5

What is HTML5?

- ✓ **HTML5** is the latest version of Hyper Text Markup Language.
- ✓ The version of this language is 5 and it is still under development.
- ✓ The main aim of HTML5 is to increase the latest multimedia support and to decrease the dependency on client side scripting language like **JavaScript** and third party plugging like '**Adobe Flash Player**' etc
- ✓ HTML5 can be effectively run on low-powered devices such as smart phones, tablets etc.

HTML5=HTML+CSS+JS

The HTML5 <!DOCTYPE>

In HTML5 there is only one <!doctype> declaration, and it is very simple:

```
<!DOCTYPE html>
```

<!DOCTYPE> is a special tag which is declared in the very beginning of a web page before <html> tag. It sends an instruction to the web browser about the version of HTML. <!DOCTYPE> is not a HTML tag and most of the modern browsers support this tag declaration.

CANVAS:

A canvas is a rectangular area on an HTML page, and it is specified with the <canvas> element.
<canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript).

<canvas> element which has only two specific attributes **width** and **height** plus all the core HTML5 attributes like id, name and class etc.

```
<canvas id="mycanvas" width="100" height="100"></canvas>
```

K Srinivas
Vasu34k@gmail.com

Canvas Coordinates

The canvas is a two-dimensional grid.

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

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.

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
- stroke()

HTML5 Canvas Examples:

- This tutorial covers following examples related to HTML5 <canvas> element.

Examples	Description
Drawing Rectangles	Learn how to draw rectangle using HTML5 <canvas> element
Drawing Paths	Learn how to make shapes using paths in HTML5 <canvas> element
Drawing Lines	Learn how to draw lines using HTML5 <canvas> element
Drawing Bezier	Learn how to draw bezier curve using HTML5 <canvas> element
Drawing Quadratic	Learn how to draw quadratic curve using HTML5 <canvas> element
Using Images	Learn how to use images with HTML5 <canvas> element
Create Gradients	Learn how to create gradients using HTML5 <canvas> element
Styles and Colors	Learn how to apply styles and colors using HTML5 <canvas> element
Text and Fonts	Learn how to draw amazing text using different fonts and their size.
Pattern and Shadow	Learn how to draw different patterns and drop shadows.
Canvas States	Learn how to save and restore canvas states while doing complex drawings on a canvas.
Canvas Translation	This method is used to move the canvas and its origin to a different point in the grid.

K Srinivas
Vasu34k@gmail.com

Canvas Rotation	This method is used to rotate the canvas around the current origin.
Canvas Scaling	This method is used to increase or decrease the units in a canvas grid.
Canvas Transform	These methods allow modifications directly to the transformation matrix.
Canvas Composition	This method is used to mask off certain areas or clear sections from the canvas.
Canvas Animation	Learn how to create basic animation using HTML5 canvas and Javascript.

Drawing Rectangles:

here are three methods that draw rectangles on the canvas:

SN	Method and Description
1	**fillRect(x,y,width,height)** This method draws a filled rectangle.
2	**strokeRect(x,y,width,height)** This method draws a rectangular outline.
3	**clearRect(x,y,width,height)** This method clears the specified area and makes it fully transparent

Here x and y specify the position on the canvas (relative to the origin) of the top-left corner of the rectangle and *width* and *height* are width and height of the rectangle.

Drawing Paths:

There are following methods required to draw paths on the canvas:

SN	Method and Description
1	**beginPath()** This method resets the current path.
2	**moveTo(x, y)** This method creates a new subpath with the given point.
3	**closePath()** This method marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath.
4	**fill()**

K Srinivas
Vasu34k@gmail.com

	This method fills the subpaths with the current fill style.
5	**stroke()** This method strokes the subpaths with the current stroke style.
6	**arc(x, y, radius, startAngle, endAngle, anticlockwise)** Adds points to the subpath such that the arc described by the circumference of the circle described by the arguments, starting at the given start angle and ending at the given end angle, going in the given direction, is added to the path, connected to the previous point by a straight line.

Drawing Lines:

There are following methods required to draw lines on the canvas:

SN	Method and Description
1	**beginPath()** This method resets the current path.
2	**moveTo(x, y)** This method creates a new subpath with the given point.
3	**closePath()** This method marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath.
4	**fill()** This method fills the subpaths with the current fill style.
5	**stroke()** This method strokes the subpaths with the current stroke style.
6	**lineTo(x, y)** This method adds the given point to the current subpath, connected to the previous one by a straight line.

Drawing Bezier:

There are following methods required to draw bezier on the canvas:

SN	Method and Description
1	**beginPath()** This method resets the current path.
2	**moveTo(x, y)** This method creates a new subpath with the given point.
3	**closePath()** This method marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath.

K Srinivas
Vasu34k@gmail.com

4	**fill()** This method fills the subpaths with the current fill style.
5	**stroke()** This method strokes the subpaths with the current stroke style.
6	**bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)** This method adds the given point to the current path, connected to the previous one by a cubic Bezier curve with the given control points.

The x and y parameters in bezierCurveTo() method are the coordinates of the end point. cp1x and cp1y are the coordinates of the first control point, and cp2x and cp2y are the coordinates of the second control point.

Drawing Quadratic:

There are following methods required to draw quadratic curve on the canvas:

SN	Method and Description
1	**beginPath()** This method resets the current path.
2	**moveTo(x, y)** This method creates a new subpath with the given point.
3	**closePath()** This method marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath.
4	**fill()** This method fills the subpaths with the current fill style.
5	**stroke()** This method strokes the subpaths with the current stroke style.
6	**quadraticCurveTo(cpx, cpy, x, y)** This method adds the given point to the current path, connected to the previous one by a quadratic Bezier curve with the given control point.

Using Images:

This tutorial would show how to import and external image into a canvas and then how to draw on that image by using following methods:

SN	Method and Description
1	**beginPath()** This method resets the current path.
2	**moveTo(x, y)** This method creates a new subpath with the given point.

K Srinivas
Vasu34k@gmail.com

3	**closePath()** This method marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath.
4	**fill()** This method fills the subpaths with the current fill style.
5	**stroke()** This method strokes the subpaths with the current stroke style.
6	**drawImage(image, dx, dy)** This method draws the given image onto the canvas. Here *image* is a reference to an image or canvas object. x and y form the coordinate on the target canvas where our image should be placed.

Create Gradients:

HTML5 canvas allows us to fill and stroke shapes using linear and radial gradients using the following methods:

SN	Method and Description
1	**addColorStop(offset, color)** This method adds a color stop with the given color to the gradient at the given offset. Here 0.0 is the offset at one end of the gradient, 1.0 is the offset at the other end.
2	**createLinearGradient(x0, y0, x1, y1)** This method returns a CanvasGradient object that represents a linear gradient that paints along the line given by the coordinates represented by the arguments. The four arguments represent the starting point (x1,y1) and end point (x2,y2) of the gradient.
3	**createRadialGradient(x0, y0, r0, x1, y1, r1)** This method returns a CanvasGradient object that represents a radial gradient that paints along the cone given by the circles represented by the arguments. The first three arguments define a circle with coordinates (x1,y1) and radius r1 and the second a circle with coordinates (x2,y2) and radius r2.

Styles and Colors

HTML5 canvas provides following two important properties to apply colors to a shape:

SN	Method and Description
1	**fillStyle** This attribute represents the color or style to use inside the shapes.
2	**strokeStyle** This attribute represents the color or style to use for the lines around shapes

K Srinivas
Vasu34k@gmail.com

By default, the stroke and fill color are set to black which is CSS color value #000000.

Text and Fonts:

TML5 canvas provides capabilities to create text using different font an dtext properties listed below:

SN	Property and Description
1	**font [= value]** This property returns the current font settings and can be set, to change the font.
2	**textAlign [= value]** This property returns the current text alignment settings and can be set, to change the alignment. The possible values are **start, end, left, right,** and **center**.
3	**textBaseline [= value]** This property returns the current baseline alignment settings and can be set, to change the baseline alignment. The possible values are **top, hanging, middle , alphabetic, ideographic** and **bottom**
4	**fillText(text, x, y [, maxWidth])** This property fills the given *text* at the given position indicated by the given coordinates x and y.
5	**strokeText(text, x, y [, maxWidth])** This property strokes the given *text* at the given position indicated by the given coordinates x and y.

Canvas States:

HTML5 canvas provides two important methods to save and restore the canvas states. The canvas drawing state is basically a snapshot of all the styles and transformations that have been applied and consists of the followings:

1. The transformations such as translate, rotate and scale etc.
2. The current clipping region.
3. The current values of the following attributes: *strokeStyle, fillStyle, globalAlpha, lineWidth, lineCap, lineJoin, miterLimit, shadowOffsetX, shadowOffsetY, shadowBlur, shadowColor, globalCompositeOperation, font, textAlign, textBaseline.*

Canvas states are stored on a stack every time the **save** method is called, and the last saved state is returned from the stack every time the **restore** method is called.

SN	Method and Description
1	**save()** This method pushes the current state onto the stack..
2	**restore()** This method pops the top state on the stack, restoring the context to that state.

K Srinivas
Vasu34k@gmail.com

Canvas Animation:

HTML5 canvas provides necessary methods to draw an image and erase it completely. We can take Javascript help to simulate good animation over a HTML5 canvas.

Following are the two important Javascript methods which would be used to animate an image on a canvas:

SN	Method and Description
1	**setInterval(callback, time);** This method repeatedly executes the supplied code after a given *time* milliseconds.
2	**setTimeout(callback, time);** This method executes the supplied code only once after a given *time* milliseconds.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function drawShape(){
  // get the canvas element using the DOM
  var canvas = document.getElementById('mycanvas');

  // Make sure we don't execute when canvas isn't supported
  if (canvas.getContext){

    // use getContext to use the canvas for drawing
    var ctx = canvas.getContext('2d');

    // Filled triangle
    ctx.beginPath();
    ctx.moveTo(25,25);
    ctx.lineTo(105,25);
    ctx.lineTo(25,105);
    ctx.fill();

    // Stroked triangle
    ctx.beginPath();
    ctx.moveTo(125,125);
    ctx.lineTo(125,45);
    ctx.lineTo(45,125);
    ctx.closePath();
    ctx.stroke();

  } else {
    alert('You need Safari or Firefox 1.5+ to see this demo.');
  }
}
</script>
</head>
<body onload="drawShape();">
   <canvas id="mycanvas"></canvas>
</body>
</html>
```

K Srinivas
Vasu34k@gmail.com

HTML5 Web storage:

HTML5 introduces two mechanisms, similar to HTTP session cookies, for storing structured data on the client side and to overcome following drawbacks.

- Cookies are included with every HTTP request, thereby slowing down your web application by transmitting the same data.
- Cookies are included with every HTTP request, thereby sending data unencrypted over the internet.
- Cookies are limited to about 4 KB of data . Not enough to store required data.

The two storages are **session storage** and **local storage** and they would be used to handle different situations.

The data is stored in key/value pairs, and a web page can only access data stored by itself.

localStorage and sessionStorage

There are two new objects for storing data on the client:

- localStorage - stores data with no expiration date
- sessionStorage - stores data for one session

The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

The sessionStorage Object

The sessionStorage object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the browser window.

Delete Web Storage:

To clear a local storage setting you would need to call localStorage.remove('key')**; where 'key' is the key of the value you want to remove. If you want to clear all settings, you need to call** localStorage.clear() **method.**

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

 <script type="text/javascript">
  if( sessionStorage.hits ){
    sessionStorage.hits = Number(sessionStorage.hits) +1;
```

K Srinivas
Vasu34k@gmail.com

```
  }else{
    sessionStorage.hits = 1;
  }
  document.write("Total Hits :" + sessionStorage.hits);
 </script>
 <p>Refresh the page to increase number of hits.</p>
</body>
</html>
```

HTML5 New Form Attributes

HTML5 has several new attributes for <form> and <input>.

New attributes for <form>:

- autocomplete
- novalidate

New attributes for <input>:

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step

<form> / <input> autocomplete Attribute

The autocomplete attribute specifies whether a form or input field should have autocomplete on or off.

When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

Tip: It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.

Note: The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

K Srinivas
Vasu34k@gmail.com

<form> novalidate Attribute

The novalidate attribute is a boolean attribute.

When present, it specifies that the form-data (input) should not be validated when submitted.

<input> autofocus Attribute

The autofocus attribute is a boolean attribute.

When present, it specifies that an <input> element should automatically get focus when the page loads.

<input> form Attribute

The form attribute specifies one or more forms an <input> element belongs to.

Tip: To refer to more than one form, use a space-separated list of form ids.

<input> formaction Attribute

The formaction attribute specifies the URL of a file that will process the input control when the form is submitted.

The formaction attribute overrides the action attribute of the <form> element.

Note: The formaction attribute is used with type="submit" and type="image".

<input> formenctype Attribute

The formenctype attribute specifies how the form-data should be encoded when submitting it to the server (only for forms with method="post")

The formenctype attribute overrides the enctype attribute of the <form> element.

Note: The formenctype attribute is used with type="submit" and type="image".

<input> formmethod Attribute

The formmethod attribute defines the HTTP method for sending form-data to the action URL.

The formmethod attribute overrides the method attribute of the <form> element.

K Srinivas
Vasu34k@gmail.com

Note: The formmethod attribute can be used with type="submit" and type="image".

<input> height and width Attributes

The height and width attributes specify the height and width of an <input> element.

Note: The height and width attributes are only used with <input type="image">.

Tip: Always specify both the height and width attributes for images. If height and width are set, the space required for the image is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the image, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the images load).

<input> list Attribute

The list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.

<input> min and max Attributes

The min and max attributes specify the minimum and maximum value for an <input> element.

Note: The min and max attributes works with the following input types: number, range, date, datetime, datetime-local, month, time and week.

<input> multiple Attribute

The multiple attribute is a boolean attribute.

When present, it specifies that the user is allowed to enter more than one value in the <input> element.

Note: The multiple attribute works with the following input types: email, and file.

<input> placeholder Attribute

The placeholder attribute specifies a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format).

The hint is displayed in the input field when it is empty, and disappears when the field gets focus.

Note: The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

K Srinivas
Vasu34k@gmail.com

<input> required Attribute

The required attribute is a boolean attribute.

When present, it specifies that an input field must be filled out before submitting the form.

Note: The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

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

<form action="demo_form.asp">
  Select a time: <input type="time" name="usr_time">
  <input type="submit">
</form>

</body>
</html>
```

HTML5 New Input Types

HTML5 has several new input types for forms. These new features allow better input control and validation.

This chapter covers the new input types:

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

K Srinivas
Vasu34k@gmail.com

HTML5 Geolocation

HTML5 Geolocation API lets you share your location with your favorite web sites. A Javascript can capture your latitude and longitude and can be sent to backend web server and do fancy location-aware things like finding local businesses or showing your location on a map.

The geolocation APIs work with a new property of the global navigator object ie. Geolocation object which can be created as follows:

```
var geolocation = navigator.geolocation;
```

Geolocation Methods:

The geolocation object provides the following methods:

Method	Description
getCurrentPosition()	This method retrieves the current geographic location of the user.
watchPosition()	This method retrieves periodic updates about the current geographic location of the device.
clearWatch()	This method cancels an ongoing watchPosition call.

The following table describes the properties of the Position object. For the optional properties if the system cannot provide a value, the value of the property is set to null.

Property	Type	Description
coords	objects	Specifies the geographic location of the device. The location is expressed as a set of geographic coordinates together with information about heading and speed.
coords.latitude	Number	Specifies the latitude estimate in decimal degrees. The value range is [-90.00, +90.00].
coords.longitude	Number	Specifies the longitude estimate in decimal degrees. The value range is [-180.00, +180.00].
coords.altitude	Number	**[Optional]** Specifies the altitude estimate in meters above the WGS 84 ellipsoid.
coords.accuracy	Number	**[Optional]** Specifies the accuracy of the latitude and longitude estimates in meters.
coords.altitudeAccuracy	Number	**[Optional]** Specifies the accuracy of the altitude estimate in meters.
coords.heading	Number	**[Optional]** Specifies the device's current direction of movement in degrees counting clockwise relative to true north.
coords.speed	Number	**[Optional]** Specifies the device's current ground speed in meters per second.

K Srinivas
Vasu34k@gmail.com

timestamp	date	Specifies the time when the location information was retrieved and the Position object created.

Handling Errors

Geolocation is complicated, and it is very much required to catch any error and handle it gracefully.

The geolocations methods getCurrentPosition() and watchPosition() make use of an error handler callback method which gives **PositionError** object. This object has following two properties:

Property	Type	Description
code	Number	Contains a numeric code for the error.
message	String	Contains a human-readable description of the error.

The following table describes the possible error codes returned in the PositionError object.

Code	Constant	Description
0	UNKNOWN_ERROR	The method failed to retrieve the location of the device due to an unknown error.
1	PERMISSION_DENIED	The method failed to retrieve the location of the device because the application does not have permission to use the Location Service.
2	POSITION_UNAVAILABLE	The location of the device could not be determined.
3	TIMEOUT	The method was unable to retrieve the location information within the specified maximum timeout interval.

Position Options:

Following is the actual syntax of getCurrentPosition() method:

```
getCurrentPosition(callback, ErrorCallback, options)
```

Here third argument is the **PositionOptions** object which specifies a set of options for retrieving the geographic location of the device.

Following are the options which can be specified as third argument:

Property	Type	Description
enableHighAccuracy	Boolean	Specifies whether the widget wants to receive the most accurate location estimate possible. By default this is false.
timeout	Number	The timeout property is the number of milliseconds your web application is willing to wait for a position.

K Srinivas
Vasu34k@gmail.com

maximumAge	Number	Specifies the expiry time in milliseconds for cached location information.

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Click the button to get your coordinates:</p>
<button onclick="getLocation()">Try It</button>
<script>
var x=document.getElementById("demo");
function getLocation()
 {
 if (navigator.geolocation)
  {
  navigator.geolocation.getCurrentPosition(showPosition);
  }
 else{x.innerHTML="Geolocation is not supported by this browser.";}
 }
function showPosition(position)
 {
 x.innerHTML="Latitude: " + position.coords.latitude +
 "<br>Longitude: " + position.coords.longitude;
 }
</script>
</body>
</html>
```

Drag and Drop:

Drag and Drop (DnD) is powerful User Interface concept which makes it easy to copy, reorder and deletion of items with the help of mouse clicks. This allows the user to click and hold the mouse button down over an element, drag it to another location, and release the mouse button to drop the element there.

Drag and Drop Events:

There are number of events which are fired during various stages of the drag and drop operation. These events are listed below:

Events	Description
dragstart	Fires when the user starts dragging of the object.
dragenter	Fired when the mouse is first moved over the target element while a drag is occuring. A listener for this event should indicate whether a drop is allowed over this location. If there are no listeners, or the listeners perform no operations, then a drop is not allowed by default.

K Srinivas
Vasu34k@gmail.com

dragover	This event is fired as the mouse is moved over an element when a drag is occuring. Much of the time, the operation that occurs during a listener will be the same as the dragenter event.
dragleave	This event is fired when the mouse leaves an element while a drag is occuring. Listeners should remove any highlighting or insertion markers used for drop feedback.
drag	Fires every time the mouse is moved while the object is being dragged.
drop	The drop event is fired on the element where the drop was occured at the end of the drag operation. A listener would be responsible for retrieving the data being dragged and inserting it at the drop location.
dragend	Fires when the user releases the mouse button while dragging an object.

The DataTransfer Object:

The event listener methods for all the drag and drop events accept **Event** object which has a readonly attribute called **dataTransfer**. The **event.dataTransfer** returns **DataTransfer**object associated with the event as follows:

```
function EnterHandler(event) {
    DataTransfer dt = event.dataTransfer;
    ..............
}
```

The *DataTransfer* object holds data about the drag and drop operation. This data can be retrieved and set in terms of various attributes associated with DataTransfer object as explained below:

S.N.	DataTransfer attrobutes and their description
1	**dataTransfer.dropEffect [= value]** • Returns the kind of operation that is currently selected. • This attribute can be set, to change the selected operation. • The possible values are **none, copy, link,** and **move**.
2	**dataTransfer.effectAllowed [= value]** • Returns the kinds of operations that are to be allowed. • This attribute can be set, to change the allowed operations. • The possible values are **none, copy, copyLink, copyMove, link, linkMove, move, all** and **uninitialized**.
3	**dataTransfer.types** Returns a DOMStringList listing the formats that were set in the dragstart event. In addition, if any files are being dragged, then one of the types will be the string "Files".
4	**dataTransfer.clearData([format])**

K Srinivas
Vasu34k@gmail.com

	Removes the data of the specified formats. Removes all data if the argument is omitted.
5	**dataTransfer.setData(format, data)** Adds the specified data.
6	**data = dataTransfer.getData(format)** Returns the specified data. If there is no such data, returns the empty string.
7	**dataTransfer.files** Returns a FileList of the files being dragged, if any.
8	**dataTransfer.setDragImage(element, x, y)** Uses the given element to update the drag feedback, replacing any previously specified feedback.
9	**dataTransfer.addElement(element)** Adds the given element to the list of elements used to render the drag feedback.

Drag and Drop Process:

Following are the steps to be carried out to implement Drag and Drop operation:

Step 1: Making an Object Draggable:

Here are steps to be taken:

- If you want to drag an element, you need to set the **draggable** attribute to **true** for that element.
- Set an event listener for **dragstart** that stores the data being dragged.
- The event listener **dragstart** will set the allowed effects (copy, move, link, or some combination).

Step 2: Dropping the Object:

To accept a drop, the drop target has to listen to at least three events.

- The **dragenter** event, which is used to determine whether or not the drop target is to accept the drop. If the drop is to be accepted, then this event has to be canceled.
- The **dragover** event, which is used to determine what feedback is to be shown to the user. If the event is canceled, then the feedback (typically the cursor) is updated based on the dropEffect attribute's value.
- Finally, the **drop** event, which allows the actual drop to be performed.

```
<!DOCTYPE HTML>
```

K Srinivas
Vasu34k@gmail.com

```
<html>
<head>
<style type="text/css">
#boxA, #boxB {
   float:left;padding:10px;margin:10px; -moz-user-select:none;
}
#boxA { background-color: #6633FF; width:75px; height:75px;  }
#boxB { background-color: #FF6699; width:150px; height:150px; }
</style>
<script type="text/javascript">
function dragStart(ev) {
   ev.dataTransfer.effectAllowed='move';
   ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
   ev.dataTransfer.setDragImage(ev.target,0,0);
   return true;
}
</script>
</head>
<body>
<center>
<h2>Drag and drop HTML5 demo</h2>
<div>Try to drag the purple box around.</div>

<div id="boxA" draggable="true"
        ondragstart="return dragStart(event)">
   <p>Drag Me</p>
</div>
<div id="boxB">Dustbin</div>
</center>
</body>
</html>
```

Web worker:

A web worker is a JavaScript running in the background, without affecting the performance of the page.

Web Workers are background scripts and they are relatively heavy-weight, and are not intended to be used in large numbers.

How Web Workers Work?

Web Workers are initialized with the URL of a JavaScript file, which contains the code the worker will execute. This code sets event listeners and communicates with the script that spawned it from the main page. Following is the simple syntax:

```
var worker = new Worker('bigLoop.js');
```

K Srinivas
Vasu34k@gmail.com

Then we can send and receive messages from the web worker.

Add an "onmessage" event listener to the web worker.

```
w.onmessage=function(event){
document.getElementById("result").innerHTML=event.data;
};
```

When the web worker posts a message, the code within the event listener is executed. The data from the web worker is stored in event.data.

```
<!DOCTYPE HTML>
<html>
<head>
<title>Big for loop</title>
  <script>
      var worker = new Worker('bigLoop.js');
      worker.onmessage = function (event) {
        alert("Completed " + event.data + "iterations" );
      };

      function sayHello(){
         alert("Hello sir...." );
      }
  </script>
</head>
<body>
   <input type="button" onclick="sayHello();" value="Say Hello"/>
</body>
</html>
```

Terminate a Web Worker

When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated.

To terminate a web worker, and free browser/computer resources, use the terminate() method:

```
w.terminate();
```

K Srinivas
Vasu34k@gmail.com

```
<!DOCTYPE HTML>
<html>
<head>
<title>Big for loop</title>
  <script>
      var worker = new Worker('bigLoop.js');
      worker.onmessage = function (event) {
        alert("Completed " + event.data + "iterations" );
      };

      function sayHello(){
         alert("Hello sir...." );
      }
  </script>
</head>
<body>
   <input type="button" onclick="sayHello();" value="Say Hello"/>
</body>
</html>
```

WebSockets:

Web Sockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers.

Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a **send()** method, and receive data from server to browser by an **onmessage**event handler.

Following is the API which creates a new WebSocket object.

```
var Socket = new WebSocket(url, [protocal] );
```

WebSocket Attributes:

Following are the attribute of WebSocket object. Assuming we created Socket object as mentioned above:

Attribute	Description
Socket.readyState	The readonly attribute **readyState** represents the state of the connection. It can have the following values: 1. A value of 0 indicates that the connection has not yet been established. 2. A value of 1 indicates that the connection is established and communication is possible. 3. A value of 2 indicates that the connection is going through the closing handshake. 4. A value of 3 indicates that the connection has been closed or could not be opened.

K Srinivas
Vasu34k@gmail.com

Socket.bufferedAmount	The readonly attribute **bufferedAmount** represents the number of bytes of UTF-8 text that have been queued using send() method.

WebSocket Events:

Following are the events associated with WebSocket object. Assuming we created Socket object as mentioned above:

Event	Event Handler	Description
open	Socket.onopen	This event occurs when socket connection is established.
message	Socket.onmessage	This event occurs when client receives data from server.
error	Socket.onerror	This event occurs when there is any error in communication.
close	Socket.onclose	This event occurs when connection is closed.

WebSocket Methods:

Following are the methods associated with WebSocket object. Assuming we created Socket object as mentioned above:

Method	Description
Socket.send()	The send(data) method transmits data using the connection.
Socket.close()	The close() method would be used to terminate any existing connection.

HTML Audio

Sounds can be played in HTML by many different methods.

Using Plug-ins

A plug-in is a small computer program that extends the standard functionality of the browser.

Plug-ins can be added to HTML pages using the <object> tag or the <embed> tag.

These tags define containers for resources (normally non-HTML resources), which, depending on the type, will either be displayed by the browsers, or by an external plug-in.

Using The <embed> Element

The <embed> tag defines a container for external (non-HTML) content.

The following code fragment should play an MP3 file embedded in a web page:

K Srinivas
Vasu34k@gmail.com

Example

```
<embed height="100" width="100" src="horse.mp3">
```

Using The <object> Element

The <object tag> tag can also define a container for external (non-HTML) content.

The following code fragment should play an MP3 file embedded in a web page:

Example

```
<object height="100" width="100" data="horse.mp3"></object>
```

Yahoo Media Player - An Easy Way to Add Audio to Your Site

The FREE **Yahoo Media Player** is definitely a favorite: You simply let Yahoo do the job of playing your songs.

It plays MP3 and a lot of other formats. You can add it to your page (or blog) with a single line of code, and easily turn your HTML page into a professional playlist:

Example

```
<a href="horse.mp3">Play Sound</a>

<script src="http://mediaplayer.yahoo.com/js"></script>
```

Using A Hyperlink

If a web page includes a hyperlink to a media file, most browsers will use a "helper application" to play the file.

The following code fragment displays a link to an MP3 file. If a user clicks on the link, the browser will launch a helper application to play the file:

Example

```
<a href="horse.mp3">Play the sound</a>
```

Embedding Audio(in html 5.0):

HTML5 supports <audio> tag which is used to embed sound content in an HTML or XHTML document as follows.

K Srinivas
Vasu34k@gmail.com

```
<audio src="foo.wav" controls autoplay>
    Your browser does not support the <audio> element.
</audio>
```

Or

```
<!DOCTYPE HTML>
<html>
<body>
   <audio controls autoplay>
       <source src="/html5/audio.ogg" type="audio/ogg" />
       <source src="/html5/audio.wav" type="audio/wav" />
       Your browser does not support the <audio> element.
   </audio>
</body>
</html>
```

Audio Attribute Specification:

The HTML5 audio tag can have a number of attributes to control the look and feel and various functionalities of the control:

Attribute	Description
autoplay	This boolean attribute if specified, the audio will automatically begin to play back as soon as it can do so without stopping to finish loading the data.
autobuffer	This boolean attribute if specified, the audio will automatically begin buffering even if it's not set to automatically play.
controls	If this attribute is present, it will allow the user to control audio playback, including volume, seeking, and pause/resume playback.
loop	This boolean attribute if specified, will allow audio automatically seek back to the start after reaching at the end.
preload	This attribute specifies that the audio will be loaded at page load, and ready to run. Ignored if autoplay is present.
src	The URL of the audio to embed. This is optional; you may instead use the <source> element within the video block to specify the video to embed

Handling Media Events:

The HTML5 audio and video tag can have a number of attributes to control various functionalities of the control using Javascript:

K Srinivas
Vasu34k@gmail.com

Event	Description
abort	This event is generated when playback is aborted.
canplay	This event is generated when enough data is available that the media can be played.
ended	This event is generated when playback completes.
error	This event is generated when an error occurs.
loadeddata	This event is generated when the first frame of the media has finished loading.
loadstart	This event is generated when loading of the media begins.
pause	This event is generated when playback is paused.
play	This event is generated when playback starts or resumes.
progress	This event is generated periodically to inform the progress of the downloading the media.
ratechange	This event is generated when the playback speed changes.
seeked	This event is generated when a seek operation completes.
seeking	This event is generated when a seek operation begins.
suspend	This event is generated when loading of the media is suspended.
volumechange	This event is generated when the audio volume changes.
waiting	This event is generated when the requested operation (such as playback) is delayed pending the completion of another operation (such as a seek).

HTML Videos

Videos can be played in HTML by many different methods.

<embed> Element

The purpose of the <embed> tag is to embed multimedia elements in HTML pages.

The following HTML fragment displays a Flash video embedded in a web page:

Example

```
<embed src="intro.swf" height="200" width="200">
```

<object> Element

The purpose of the <object> tag is to embed multimedia elements in HTML pages.

The following HTML fragment displays a Flash video embedded in a web page:

K Srinivas
Vasu34k@gmail.com

Example

```
<object data="intro.swf" height="200" width="200">
```

Using the HTML5 <video> Element

The HTML5 <video> tag defines a video or movie.

The <video> element works in all modern browsers.

The following HTML fragment displays a video in OGG, MP4, or WEBM format:

Example

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

Using A Hyperlink

If a web page includes a hyperlink to a media file, most browsers will use a "helper application" to play the file.

The following code fragment displays a link to a Flash video. If a user clicks on the link, the browser will launch a helper application to play the file:

Example

```
<a href="intro.swf">Play a video file</a>
```

YouTube Embedded

```
<embed
width="420" height="345"
src="http://www.youtube.com/v/XGSy3_Czz8k"
type="application/x-shockwave-flash">
</embed>
```

<keygen> Tag

The <keygen> tag is intended to be used in a form.

K Srinivas
Vasu34k@gmail.com

This tag is provided to generate keys and to submit the public keys as part of HTML form. The private key is encrypted and stored in the local key database while the public key is packaged and sent to the server for authentication.

Ex:

```
<keygen name="security">
```

HTML5 Inline SVG

SVG stands for **S**calable **V**ector **G**raphics and it is a language for describing 2D-graphics and graphical applications in XML format.

SVG is mostly useful for vector type diagrams like Pie charts, Two-dimensional graphs in an X,Y coordinate system etc.

Embeding SVG in HTML5

HTML5 allows embeding SVG directly using **<svg>...</svg>** tag which has following simple syntax:

```
<svg xmlns="http://www.w3.org/2000/svg">
...
</svg>
```

HTML5 - SVG Circle

Following is the HTML5 version of an SVG example which would draw a cricle using <circle> tag:

```
<!DOCTYPE html>
<head>
<title>SVG</title>
<meta charset="utf-8" />
</head>
<body>
<h2>HTML5 SVG Circle</h2>
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
    <circle id="redcircle" cx="50" cy="50" r="50" fill="red" />
</svg>
</body>
</html>
```

Comparison of Canvas and SVG

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

K Srinivas
Vasu34k@gmail.com

Canvas	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	• 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

HTML5 Server-Sent Events:

HTML5 Server-Sent Events allow a web page to get updates from a server.

Server-Sent Events - One Way Messaging

A server-sent event is when a web page automatically gets updates from a server.

This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

Receive Server-Sent Event Notifications

The EventSource object is used to receive server-sent event notifications:

Example

```
var source=new EventSource("demo_sse.php");
source.onmessage=function(event)
  {
  document.getElementById("result").innerHTML+=event.data + "<br>";
  };
```

The EventSource Object

In the examples above we used the onmessage event to get messages. But other events are also available:

Events	Description

K Srinivas
Vasu34k@gmail.com

onopen	When a connection to the server is opened
onmessage	When a message is received
onerror	When an error occurs

```
<!DOCTYPE html>
<html>
<body>
<h1>Getting server updates</h1>
<div id="result"></div>

<script>
if(typeof(EventSource)!=="undefined")
 {
 var source=new EventSource("demo_sse.php");
 source.onmessage=function(event)
  {
  document.getElementById("result").innerHTML+=event.data + "<br>";
  };
 }
else
 {
 document.getElementById("result").innerHTML="Sorry, your browser does not support
server-sent events...";
 }
</script>

</body>
</html>
```

Web Sockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers.

Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a **send()** method, and receive data from server to browser by an **onmessage**event handler.

Following is the API which creates a new WebSocket object.

```
var Socket = new WebSocket(url, [protocal] );
```

Here first argument, url, specifies the URL to which to connect. The second attribute, protocol is optional, and if present, specifies a sub-protocol that the server must support for the connection to be successful.

K Srinivas
Vasu34k@gmail.com

WebSocket Attributes:

Following are the attribute of WebSocket object. Assuming we created Socket object as mentioned above:

Attribute	Description
Socket.readyState	The readonly attribute **readyState** represents the state of the connection. It can have the following values: 1. A value of 0 indicates that the connection has not yet been established. 2. A value of 1 indicates that the connection is established and communication is possible. 3. A value of 2 indicates that the connection is going through the closing handshake. 4. A value of 3 indicates that the connection has been closed or could not be opened.
Socket.bufferedAmount	The readonly attribute **bufferedAmount** represents the number of bytes of UTF-8 text that have been queued using send() method.

WebSocket Events:

Following are the events associated with WebSocket object. Assuming we created Socket object as mentioned above:

Event	Event Handler	Description
open	Socket.onopen	This event occurs when socket connection is established.
message	Socket.onmessage	This event occurs when client receives data from server.
error	Socket.onerror	This event occurs when there is any error in communication.
close	Socket.onclose	This event occurs when connection is closed.

WebSocket Methods:

Following are the methods associated with WebSocket object. Assuming we created Socket object as mentioned above:

Method	Description

K Srinivas
Vasu34k@gmail.com

Socket.send()	The send(data) method transmits data using the connection.
Socket.close()	The close() method would be used to terminate any existing connection.

WebSocket Example:

A WebSocket is a standard bidirectional TCP socket between the client and the server. The socket starts out as a HTTP connection and then "Upgrades" to a TCP socket after a HTTP handshake. After the handshake, either side can send data.

Client Side HTML & JavaScript Code:

At the time of writing this tutorial, there are only few web browsers supporting WebSocket() interface. You can try following example with latest version of Chrome, Mozilla, Opera and Safari.

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function WebSocketTest()
{
  if ("WebSocket" in window)
  {
     alert("WebSocket is supported by your Browser!");
     // Let us open a web socket
     var ws = new WebSocket("ws://localhost:9998/echo");
     ws.onopen = function()
     {
        // Web Socket is connected, send data using send()
        ws.send("Message to send");
        alert("Message is sent...");
     };
     ws.onmessage = function (evt)
     {
        var received_msg = evt.data;
        alert("Message is received...");
     };
     ws.onclose = function()
     {
        // websocket is closed.
        alert("Connection is closed...");
     };
  }
  else
  {
     // The browser doesn't support WebSocket
     alert("WebSocket NOT supported by your Browser!");
  }
}
</script>
</head>
```

K Srinivas
Vasu34k@gmail.com

```
<body>
<div id="sse">
   <a href="javascript:WebSocketTest()">Run WebSocket</a>
</div>
</body>
</html>
```

What is Application Cache?

HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection.

Application cache gives an application three advantages:

1. Offline browsing - users can use the application when they're offline
2. Speed - cached resources load faster
3. Reduced server load - the browser will only download updated/changed resources from the server

Cache Manifest Basics

To enable application cache, include the manifest attribute in the document's <html> tag:

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

Every page with the manifest attribute specified will be cached when the user visits it. If the manifest attribute is not specified, the page will not be cached (unless the page is specified directly in the manifest file).

The recommended file extension for manifest files is: ".appcache"

The Manifest File

The manifest file is a simple text file, which tells the browser what to cache (and what to never cache).

The manifest file has three sections:

- **CACHE MANIFEST** - Files listed under this header will be cached after they are downloaded for the first time
- **NETWORK** - Files listed under this header require a connection to the server, and will never be cached
- **FALLBACK** - Files listed under this header specifies fallback pages if a page is inaccessible

K Srinivas
Vasu34k@gmail.com

CACHE MANIFEST

The first line, CACHE MANIFEST, is required:

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js
```

The manifest file above lists three resources: a CSS file, a GIF image, and a JavaScript file. When the manifest file is loaded, the browser will download the three files from the root directory of the web site. Then, whenever the user is not connected to the internet, the resources will still be available.

NETWORK

The NETWORK section below specifies that the file "login.asp" should never be cached, and will not be available offline:

```
NETWORK:
login.asp
```

An asterisk can be used to indicate that all other resources/files require an internet connection:

```
NETWORK:
*
```

FALLBACK

The FALLBACK section below specifies that "offline.html" will be served in place of all files in the /html/ catalog, in case an internet connection cannot be established:

```
FALLBACK:
/html/ /offline.html
```

Updating the Cache

Once an application is cached, it remains cached until one of the following happens:

- The user clears the browser's cache
- The manifest file is modified (see tip below)
- The application cache is programmatically updated

K Srinivas
Vasu34k@gmail.com

Notes on Application Cache

Be careful with what you cache.

Once a file is cached, the browser will continue to show the cached version, even if you change the file on the server. To ensure the browser updates the cache, you need to change the manifest file.

Note: Browsers may have different size limits for cached data (some browsers have a 5MB limit per site).