Module – 2
HTML and HTML-5

Topics covered in this module are

- • HTML URL & • Encode HTML
- • XHTML
- • HTML5 New Elements & • HTML5 Semantic.
- • HTML5 Input Types
- • HTML5 Form Elements
- • HTML5 Form Attributes
- • HTML5 Canvas
- • HTML5 SVG
- • HTML5 Video and Audio
- • HTML5 Geolocation
- • HTML5 Drag/Drop
- • HTML5 Web Storage
- • HTML5 App Cache
- • HTML5 Web Workers
- • HTML5 SSE
- • HTML Media
- • HTML Plug-ins

HTML URL.

- URL Stands for Uniform Resource Locator.
- User can given either the website address or ip address of the URL can be given.
- From the browser, user can type http://www.google.com or can type ip address of the domain google.com.
  - How do you find the ip address of a domain name?
    - In windows, go to the command prompt by typing cmd in start window.
    - In the command prompt, type “ping www.google.com”. You can see the ip address of google.com in the command prompt. (Ping will communicate with the server and returns its ip address and the time to communicate with the server. Here the server is google.com)
- Open the browser and type ip address to see the same result as typing in the domain name to connect with the server.

HTML URL.

- URL should have the following information
    - Scheme: //host.domain serve name:port/path/filename.

  Here Scheme is http / https / ftp.

    - Http stands hypertext transfer protocol.
    - Https stands hypertext transfer protocol with security. Https is used when the information between and browser and server needs to be secured. For example, credit card payment in shopping sites, internet banking etc.,
    - Ftp – FTP stands for File transfer protocol. To transfer file from one system to another system.

  - Host is www --> World wide web.
  - Domain server name --> domain name of the server. For google it is google.com.
  - Port : Defines the port at which the browser can communicate with the server.
  - Path : Specifies the program to be executed on the server. If this is not given then root of the server is assumed.
  - File name – Name of the file under the given path to be executed. File name could be html file, jsp file, asp file etc.,

Encode URL.

- For example if we want to call the following URL:

  http://www.myapp.com/new pricing.htm

  It is not possible to give the spaces in the URL as it is invalid. Rather, URL Encode format can be used to resolve this issue. %20 is the space character. The URL can be modified as below

  - http://www.myapp.com/new%20pricing.htm
- At the server side, it understands “new pricing.htm”.

XHTML.

- XHTML stands for Extensible Hypertext Markup language.
- XML and HTML are combined in XHTML.
- XHTML is a strict version of HTML.
- Rules of XHTML
  - DOCTYPE, HTML, HEADER, TITLE and BODY tags are mandatory in XHTML.
  - XHTML elements should be properly nested.
  - All the tags/elements should be lowercase.
  - All the tags/elements and end tags.elements should be properly closed.
  - XHTML should have one root element.
  - Similarly, All the attribute data should be placed in double quotes.
  - All the attributes should be in lowercase.

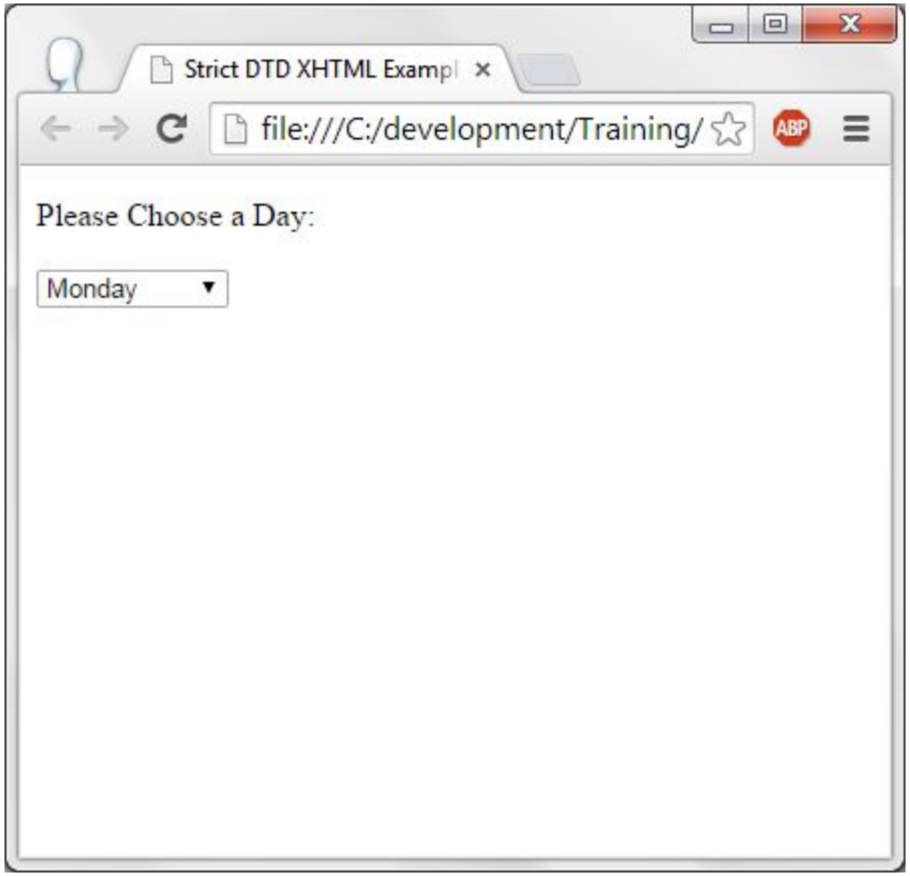
XHTML.

- Advantages of XHTML:
  - Sustainability – As the industry is moving towards XML, since XHTML also has XML, it is widely accepted.
  - Wide range of applicaton – With XHTML more complex web sites can be designed as it has data and UI.
  - Compatibility – Since XHTML has the format of XML, developers can read/parse XHTML documents and can convert it into PDF, RTF formats.
  - Efficiency of processing: Since XHTML is strict version of HTML, less number of errors will be there in the document and speed of processing the XHTML document is faster.
  - A clean code – Since XHTML has mandatory closing tags unlike HTML, it is always a clean code. For a beginner it is easy to read and understand and for programmers it is a good and clean code.

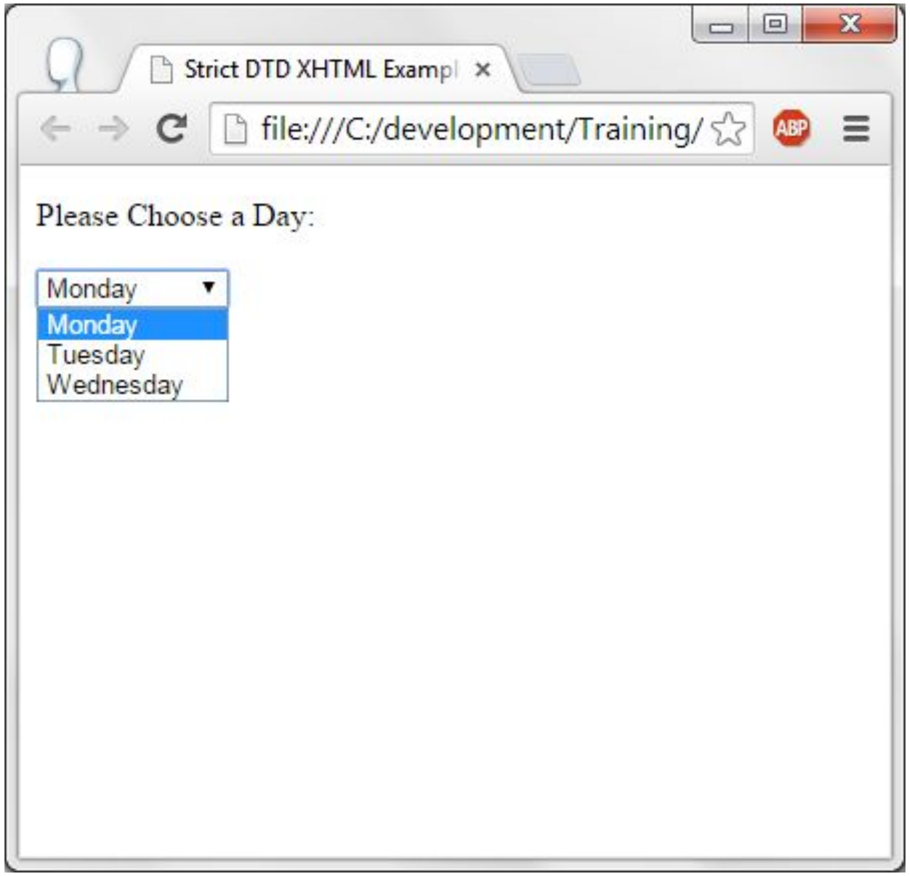
Example of XHTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title> Strict DTD XHTML Example </title>

</head>

<body>

<p>

Please Choose a Day:

<br /><br />

<select name="day">

<option selected="selected">Monday</option>

<option>Tuesday</option>

<option>Wednesday</option>

</select>

</p>

</body>

</html>
```

Output of XHTML.



Please Choose a Day:



Please Choose a Day:

New elements of HTML - 5

- New elements for better document structure are:
  - Article, header, figcaption, header, footer, section, summary etc.,
- Form elements
  - Datalist, keygen, output etc.,
- Form Input types:
  - Date, datetime, color, email, telephone, time, url etc.,
- Graphics :
  - SVG, Canvas
- Media elements:
  - Audio, video, embed, source, track etc.,

Semantic elements.

- Sementic elements are tags with a meaning. Some of the semantic elements are: <table> <img> <form>. Non-semantic elements are : div, span etc.,
- Newly introduced below given HTML-5 sementic elements does nothing except for better documentation.
  - Article – defines an article.
  - Aside – like a side bar.
  - Figure – to display set of figures.
  - Header – Section for header.
  - Foote – Section for footer.
  - Summary – Section to display summary.
- Sementic elements which does some operation are:
  - Figcaption : This displays the caption for the figure.
  - Mark – It is the highlighter.

<fig-caption> semantic tag.

```
<figure>
  <img src="roses.jpg" alt="Couple
    of roses" width="304"
    height="228">
  <figcaption>Fig1. - A view of
    Roses.</figcaption>
</figure>
```

In the above code, figcaption is used to display the caption for the image.

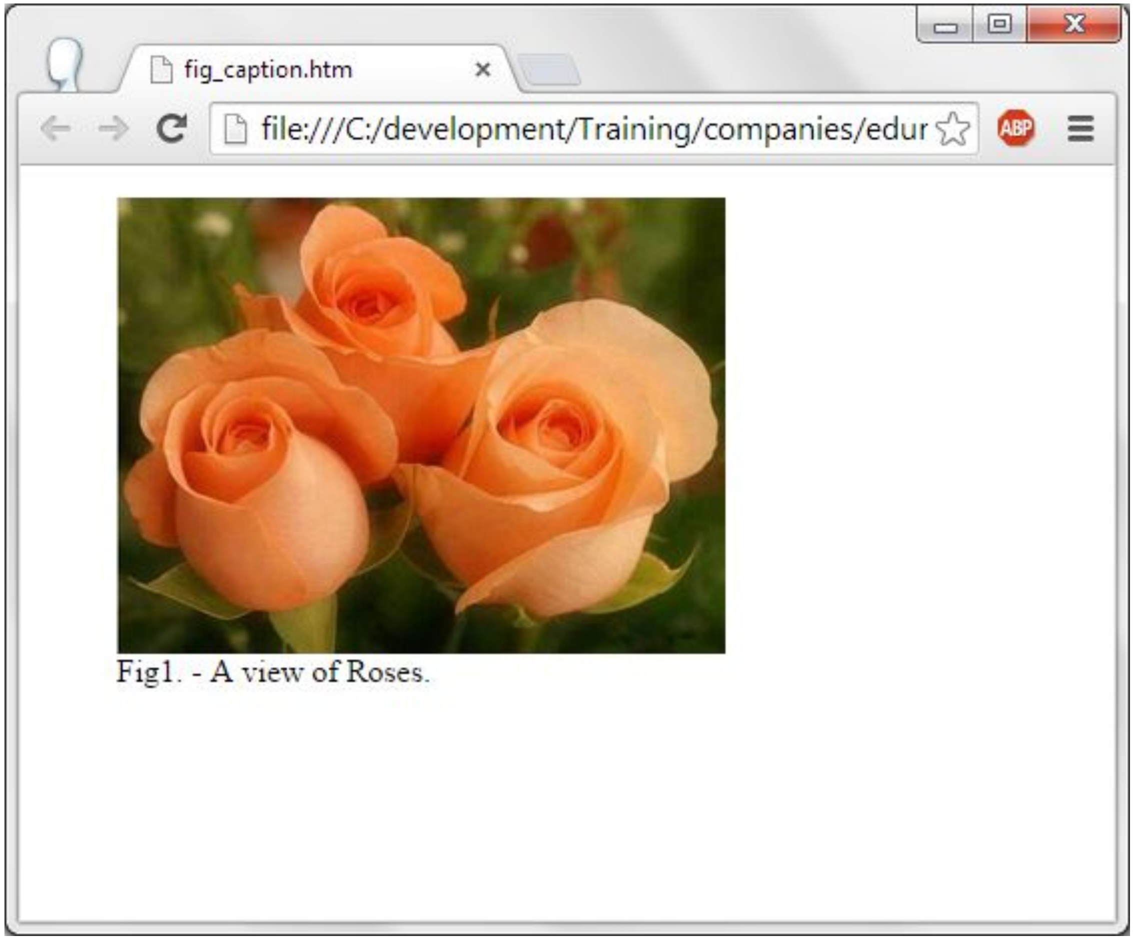


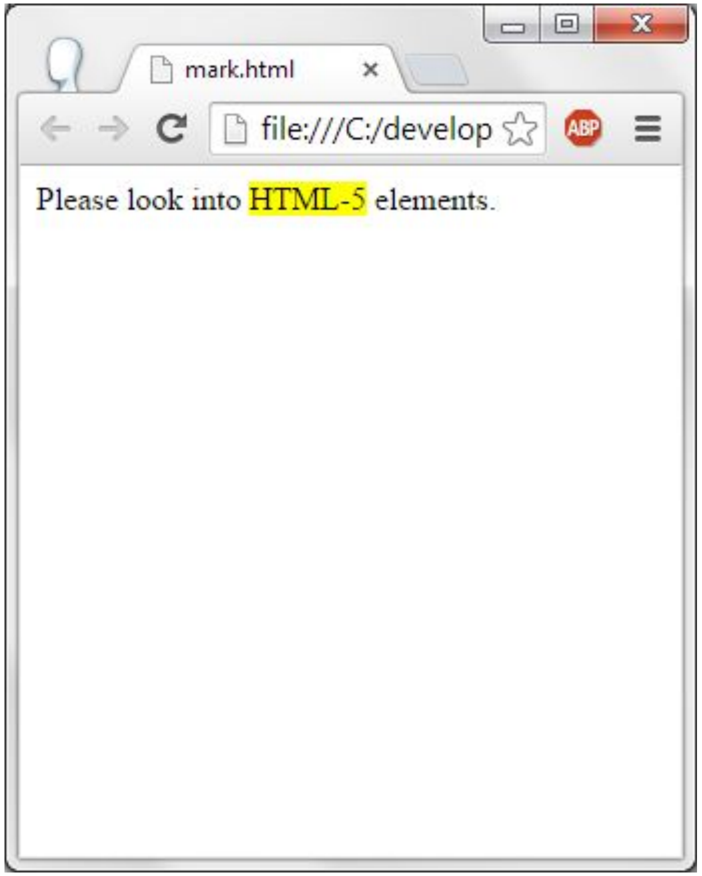
Fig1. - A view of Roses.

<mark> semantic tag.

It is the same as mark in html attribute but in html-5 it is made as a tag for semantics.

Following code using mark tag:

<p>Please look into <mark>HTML-5</mark> elements.</p>



Please look into HTML-5 elements.

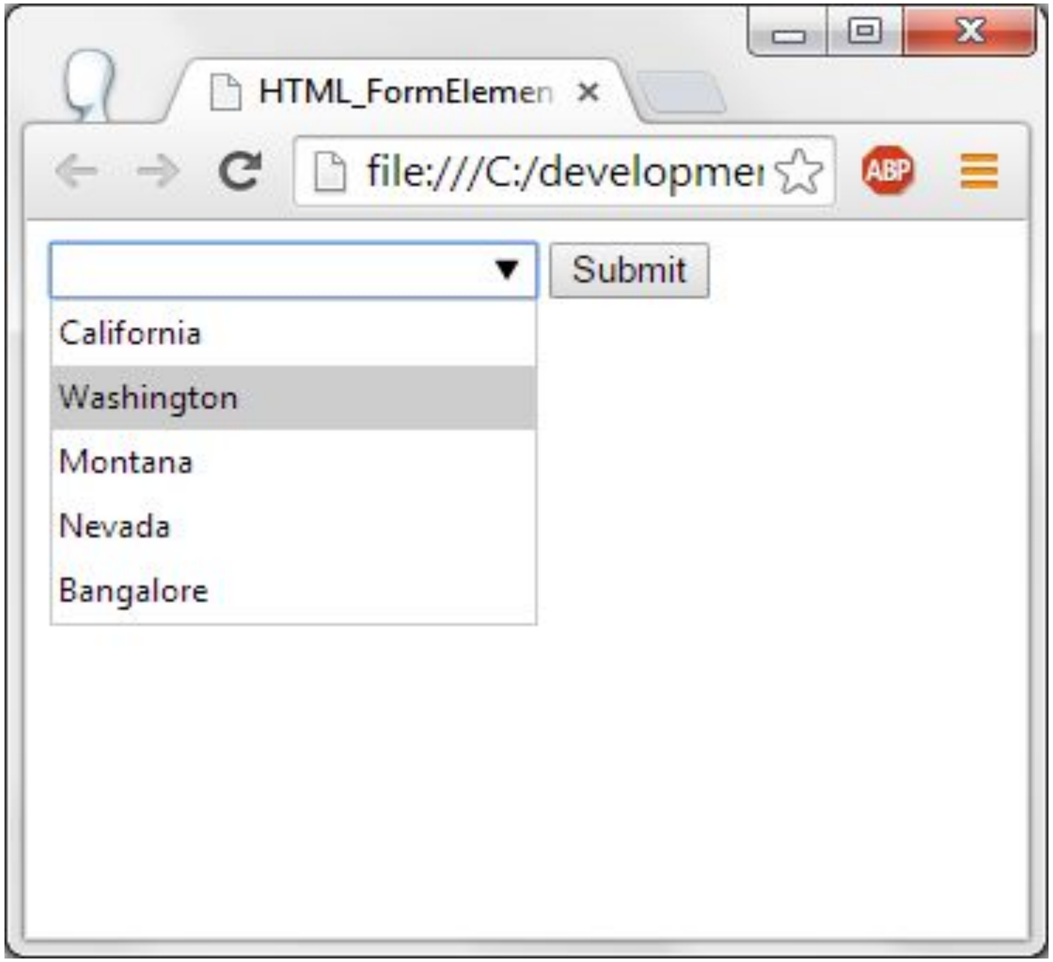
HTML5 – Form Elements

- New Form elements are <datalist> and <output>
- Datalist tag - Can define the elements for drop down for the user to select.
- Output tag – Performs the calculation and displays the result in Output element.

Form Element – Data List

```
<!DOCTYPE html>
<html>
<body>
<form action="" method="get">
<input list="State" name="State">
<datalist id="State">
  <option value="California">
  <option value="Washington">
  <option value="Montana">
  <option value="Nevada">
  <option value="Bangalore">
</datalist>
<input type="submit">
</form>
<p>Example of datalist</p>
</body>
</html>
```

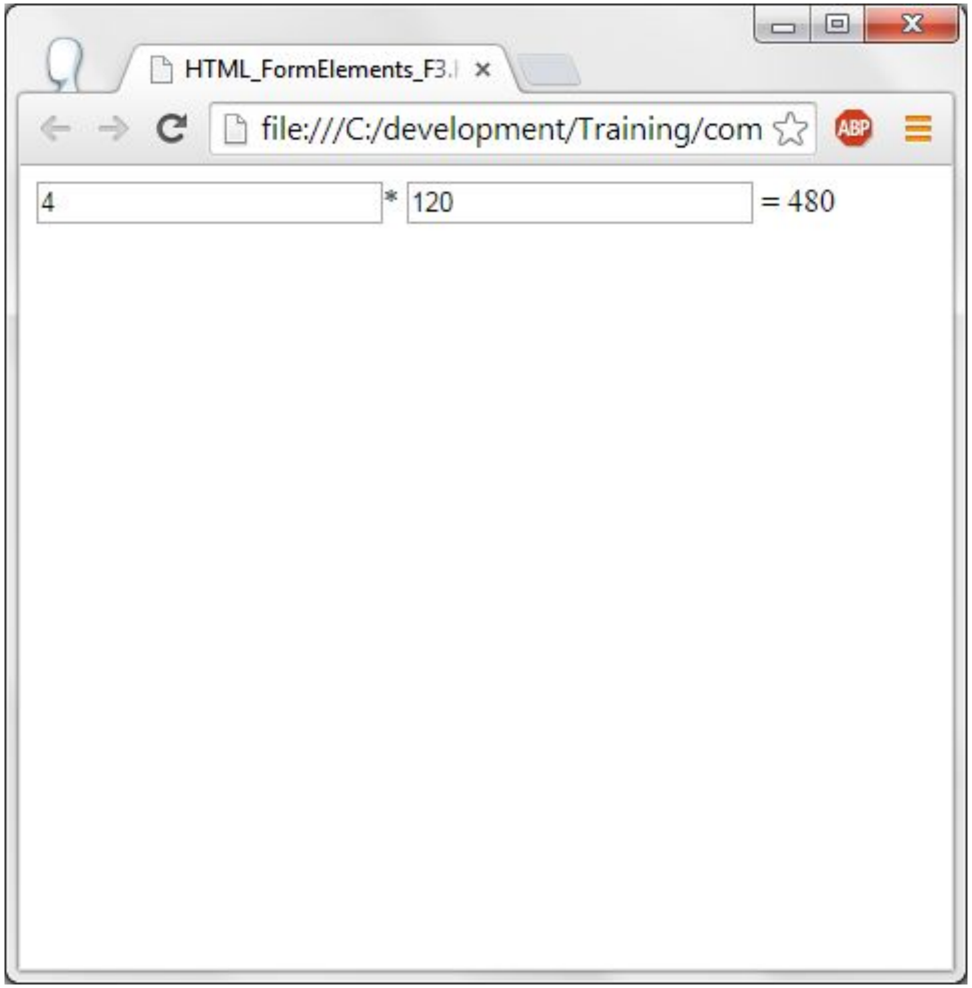
Here datalist tag provides the data for the input list state.



HTML5 - Output

- <!DOCTYPE html>
- <html><body>
- <form action="demo_form.asp" method="get"
- oninput="output.value=parseInt(first.value)*parseInt(second.value)">
- <input type="number" id="first" name="first" value="50">*
- <input type="number" id="second" name="second" value="50">=
- <output name="output" for="first second"></output>
- </form></body></html>

- Here two input numbers are taken using input tag.
- On entering data, they are converted to integer data and multiplied and displayed in output tag.



4 * 120 = 480

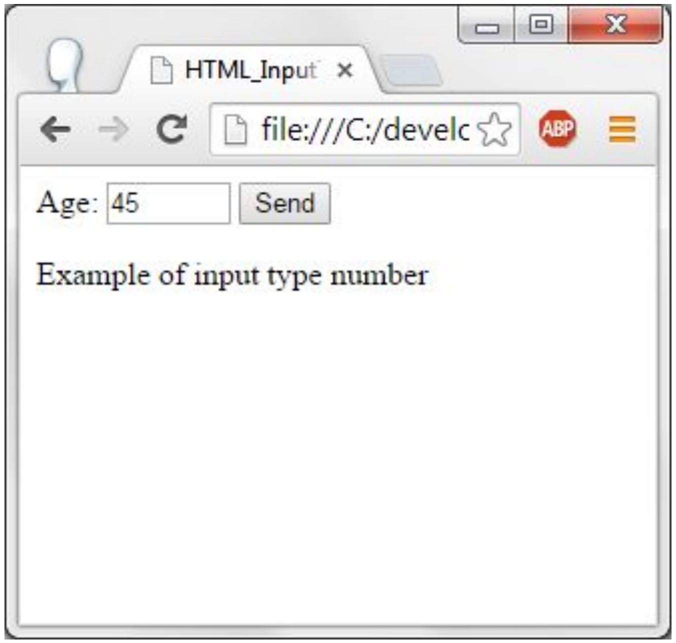
HTML5 – Input types.

- In HTML5, various input types are introduced. They are:
  - Color
  - Date
  - Datetime
  - Email
  - Url
  - Week
  - Number

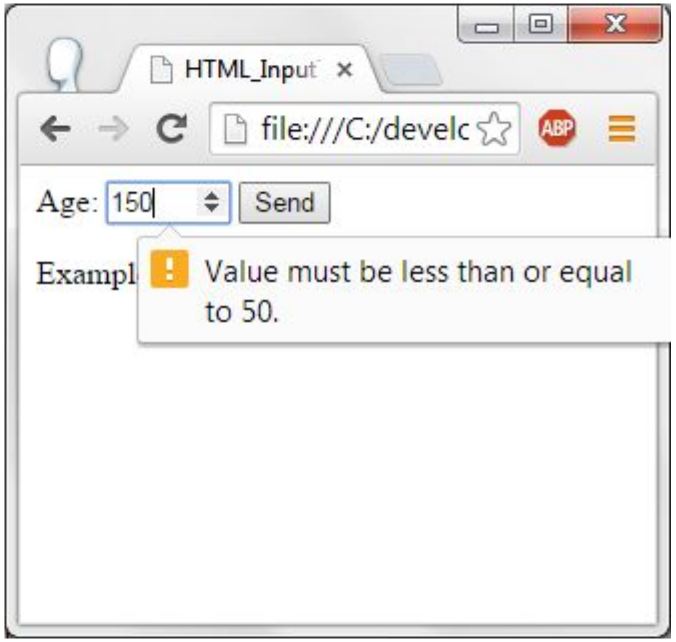
Input types - Number

```
<!DOCTYPE html>

<html><body>

<form action="">

  Age:

  <input type="number" name="age" min="1"
       max="50">

  <input type="submit" value="Send">

</form>

<p>Example of input type number</p>

</body></html>
```

- Here the input type is number. User can enter a number. Minimum and maximum limit can be specified. If the user does not enter in this bounds then message will be displayed to the user.



Example of input type number

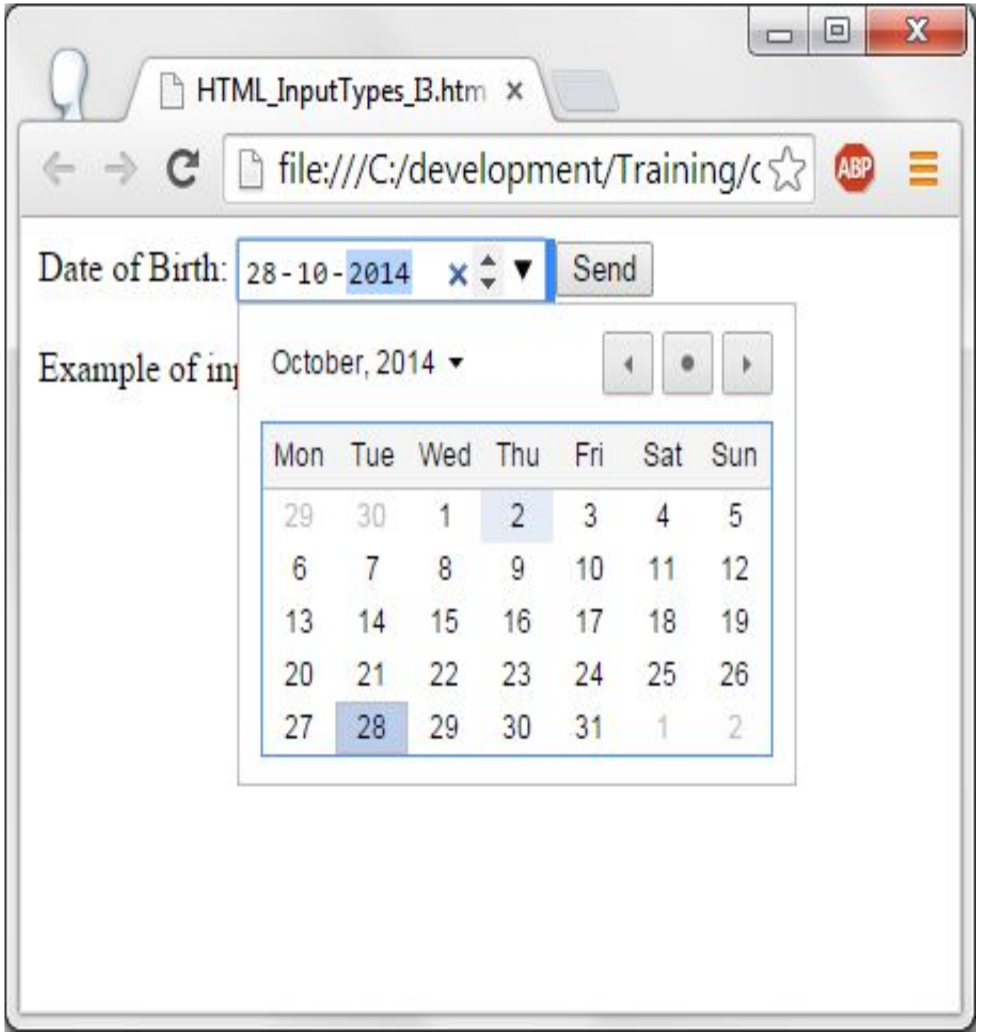


Value must be less than or equal to 50.

Input types - date

```
<!DOCTYPE html>

<html><body>

<form action="">

  Date of Birth:

  <input type="date" name="birthday">

  <input type="submit" value="Send">

</form>

<p>Example of input date</p>

</body></html>
```

- Here the input type date is used. Current Year and Month with the dates are displayed as seen in the output window.

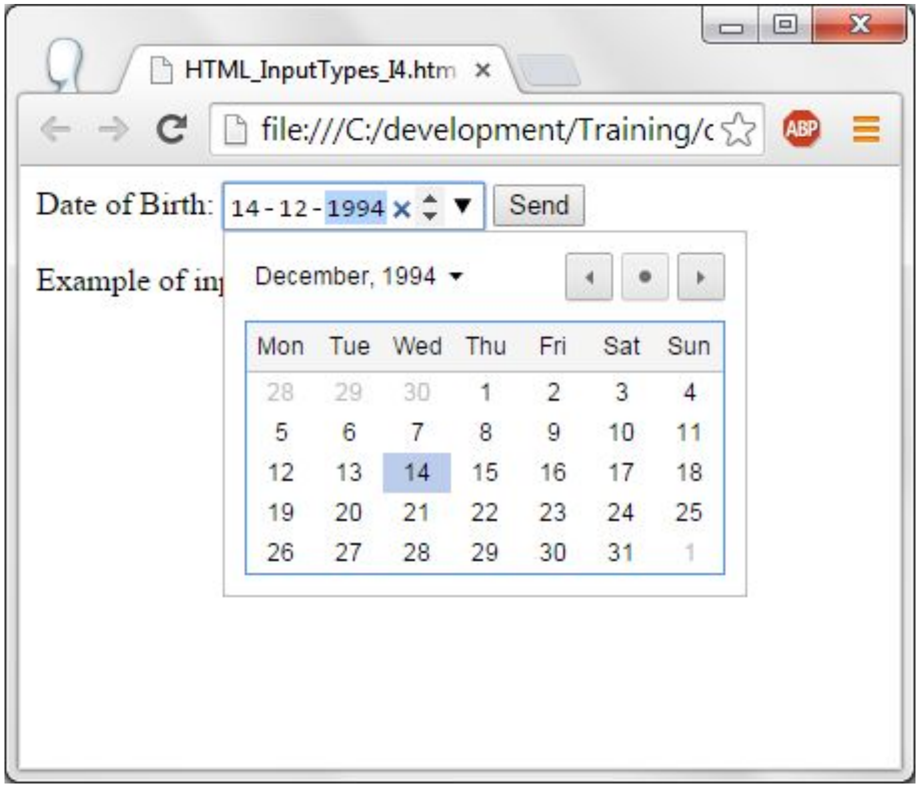


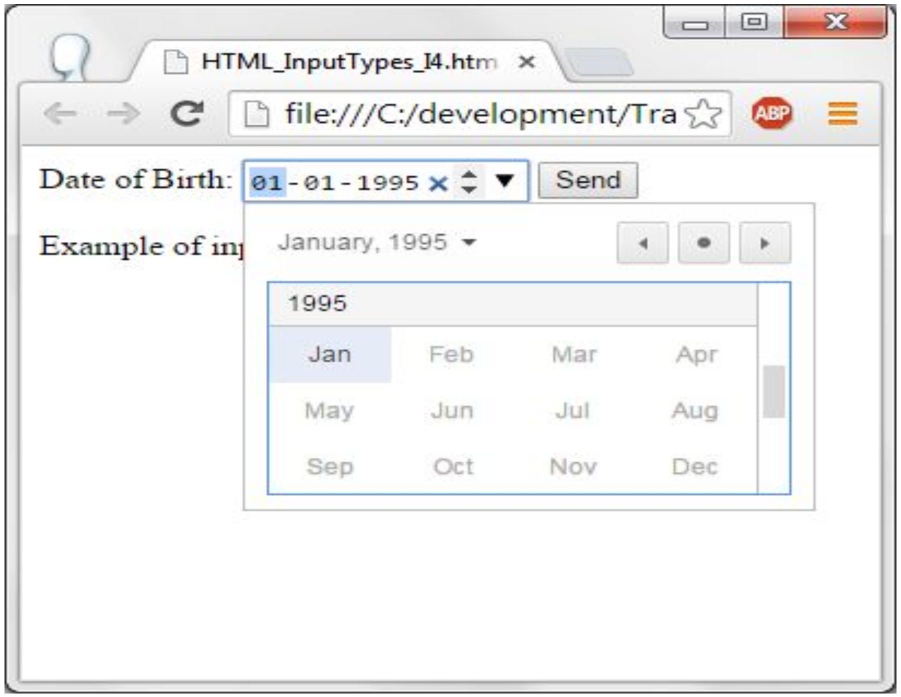
Mon	Tue	Wed	Thu	Fri	Sat	Sun
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

Input – Date - range

```
<!DOCTYPE html>

<html><body>

<form action="">

  Date of Birth:

  <input type="date" name="birthday" min="1985-01-01"
       max="1995-01-01">

  <input type="submit" value="Send">

</form>

<p>Example of input date with restriction</p>

</body></html>
```

In this code, user can select date range between min and max attributes.

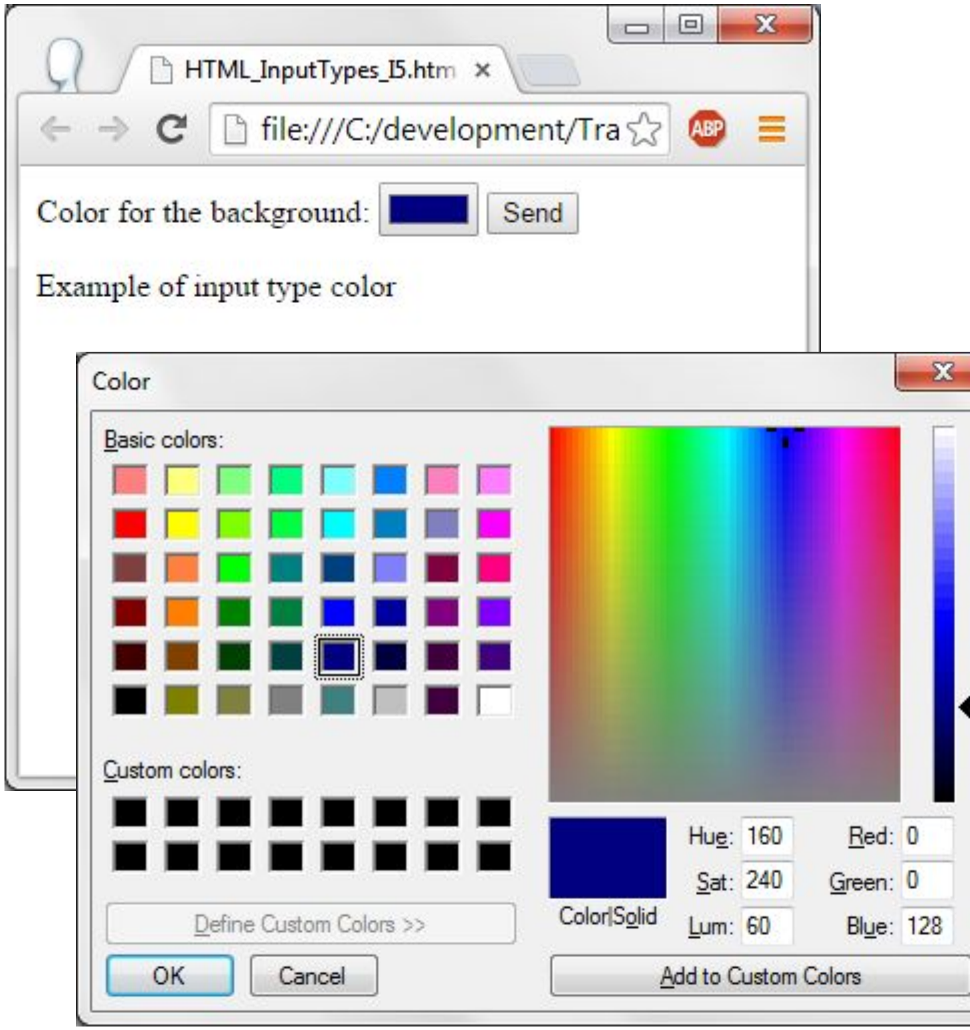




HTML5 – Input type - Color

```
<!DOCTYPE html>

<html>

<body>

<form action="">

  Color for the background:

  <input type="color" name="color" >

  <input type="submit" value="Send">

</form>

<p>Example of input type color</p>

</body>

</html>
```

Input type color is used. When the user clicks on the color button, color window is displayed. User can choose the colors in the color window and can click on ok button to select the color.



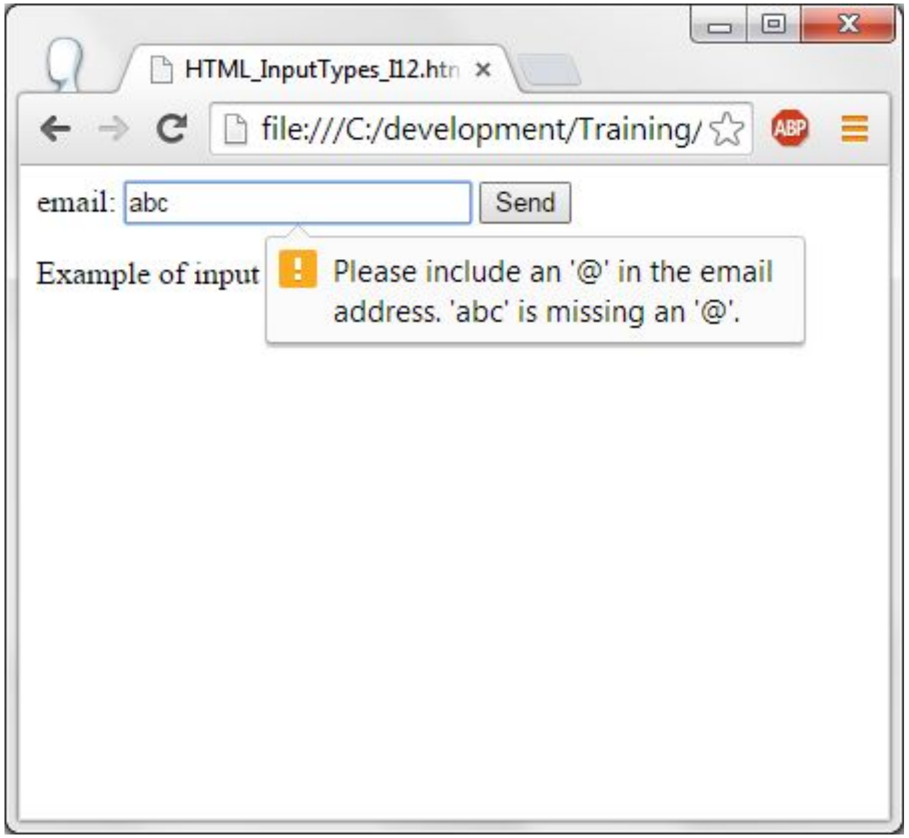
Color for the background:

Example of input type color

Hue: 160
Sat: 240
Lum: 60
Red: 0
Green: 0
Blue: 128

Input type - email

```
<!DOCTYPE html>

<html><body>

<form action="">

  email:

  <input type="email"
     name="email">

  <input type="submit" value="Send">

</form>

<p>Example of input type email</p>

</body></html>
```



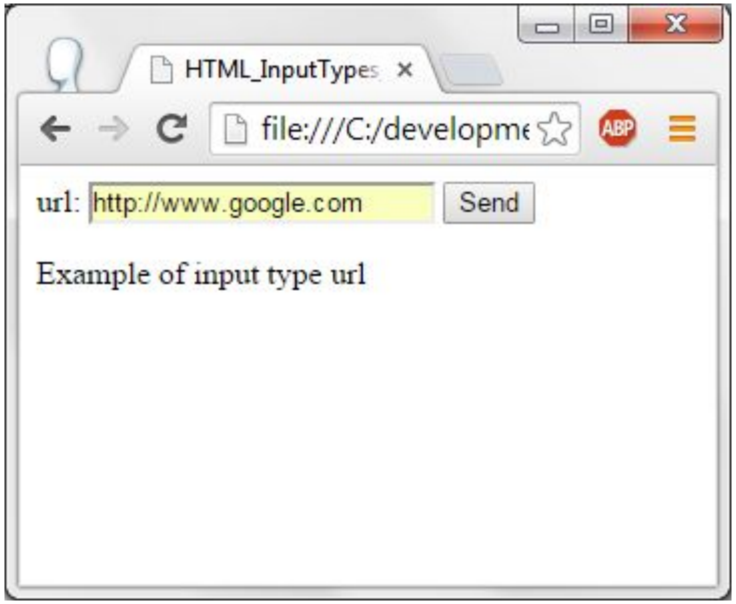
Please include an '@' in the email address. 'abc' is missing an '@'.

- Here the input type email is used. If ther user types invalid email id then HTML5 will display an error message.

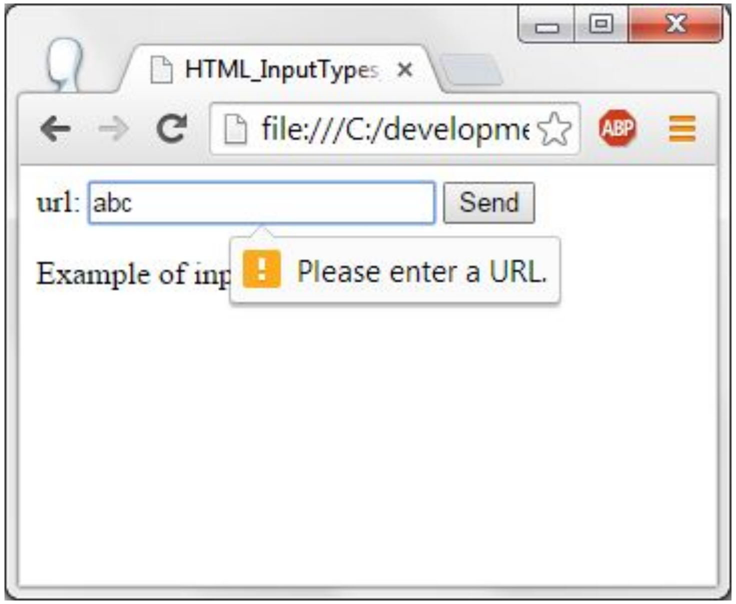
Input type - URL

```
<!DOCTYPE html>
<html><body>
<form action="">
  url:
    <input type="url" name="url">
  <input type="submit" value="Send">
</form>
<p>Example of input type url</p>
</body></html>
```

- Input type URL is used. If the user does not eneter a valid url then error message is displayed by browser.



Example of input type url



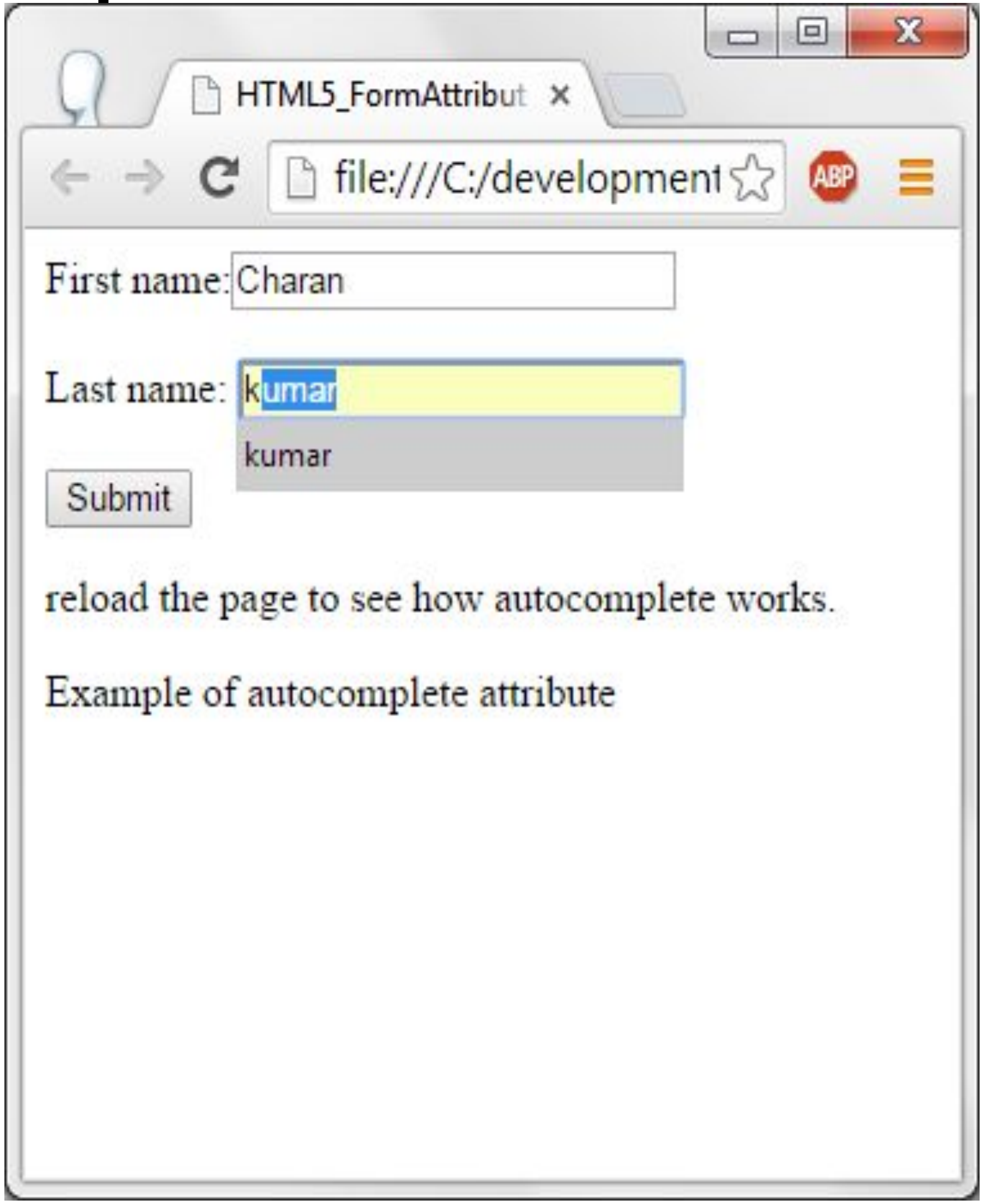
Example of inp

Please enter a URL.

HTML5 - Form Attributes - Autocomplete

```
<!DOCTYPE html>

<html><body>

<form action="" >

  First name:<input type="text" name="fname"
      autocomplete="off"/><br>

  Last name: <input type="text" name="lname"
      autocomplete="on"/><br>

  <input type="submit"/>

</form>

<p>reload the page to see how autocomplete works.</p>

<p>Example of autocomplete attribute</p>

</body></html>
```

- With the autocomplete on, user will be suggested with the option which were entered previously. First input field autocomplete is off and for the second input field – Auto complete is on.



First name: Charan

Last name: kumar

kumar

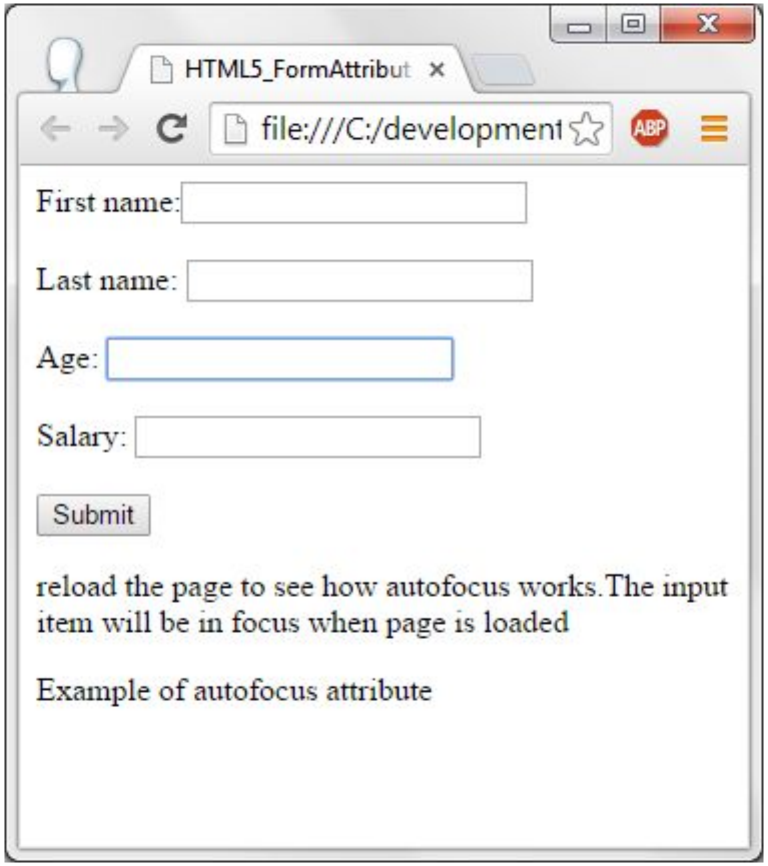
reload the page to see how autocomplete works.

Example of autocomplete attribute

Form Attribute – Autofocus

```
<!DOCTYPE html>

<html><body>

<form action="" >

  First name:<input type="text" name="fname" /><br/><br/>

  Last name: <input type="text" name="lname" /><br/><br/>

  Age: <input type="text" name="age"
      autofocus/><br/><br/>

  Salary: <input type="text" name="salary" /><br/><br/>

  <input type="submit"/>

</form>

<p>reload the page to see how autofocus works.The input item
      will be in focus when page is loaded</p>

<p>Example of autofocus attribute</p>

</body></html>
```

- Using the attribute autofocus, when the form is displayed, control will be in auto focus field first which is age field in this example.



First name:

Last name:

Age:

Salary:

Submit

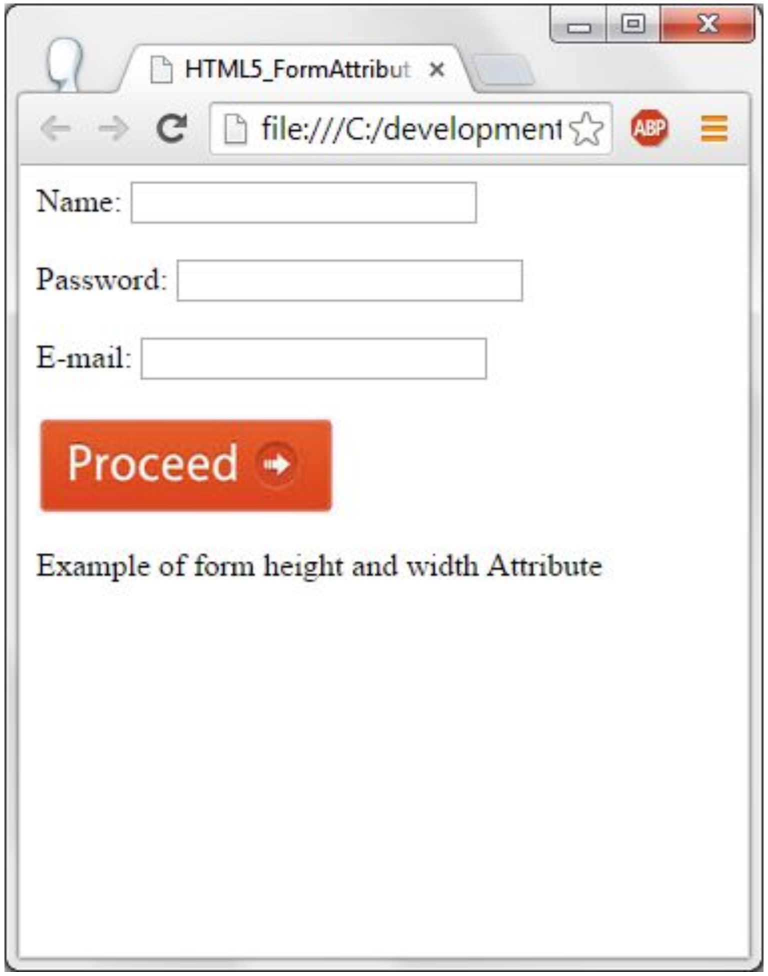
reload the page to see how autofocus works.The input item will be in focus when page is loaded

Example of autofocus attribute

Form Attribute – Height and Width.

```
<html>    <body>

      <form action="ab.php" method="post">

          <label for="name">Name:</label>

          <input type="text" name="name" id="name"
        /><br/><br/>

          <label for="mail">Password:</label>

          <input type="password" name="pwd" id="pwd"
        /><br/><br/>

          <label for="mail">E-mail:</label>

           <input type="email" id="mail" name="mail"
        /><br/><br/>

          <input type="image" height="50"
        width="150" value="Default"
        src="FormAttribute_Image.jpg" class="btn" />

         <p>Example of form height and width Attribute</p>

      </form>   </body></html>
```

- Here height and width can be specified for the input element.



Name:

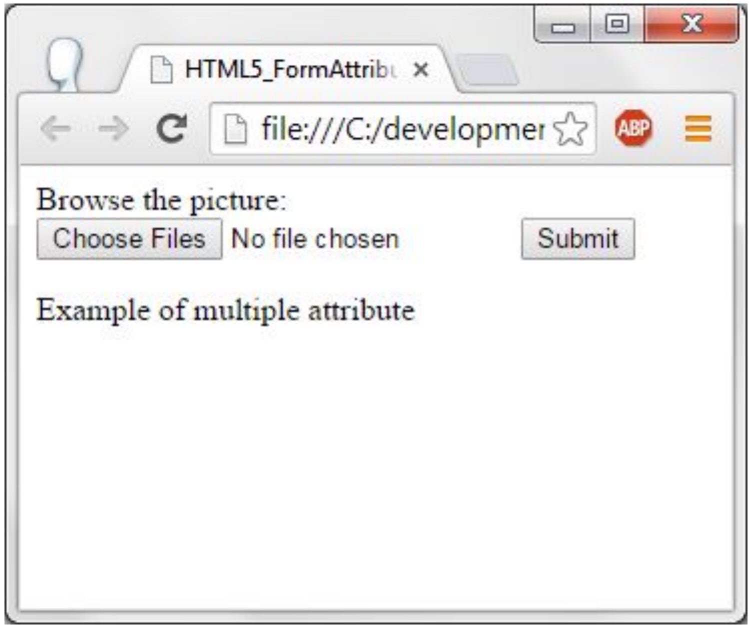
Password:

E-mail:

Example of form height and width Attribute

Form Attribute – Multiple images.

```
<!DOCTYPE html>

<html><body>

<form action="">

  Browse the picture: <input type="file" name="image"
        multiple>

  <input type="submit">

</form>

<p>Example of multiple attribute</p>

</body></html>
```



Browse the picture:

Choose Files No file chosen Submit

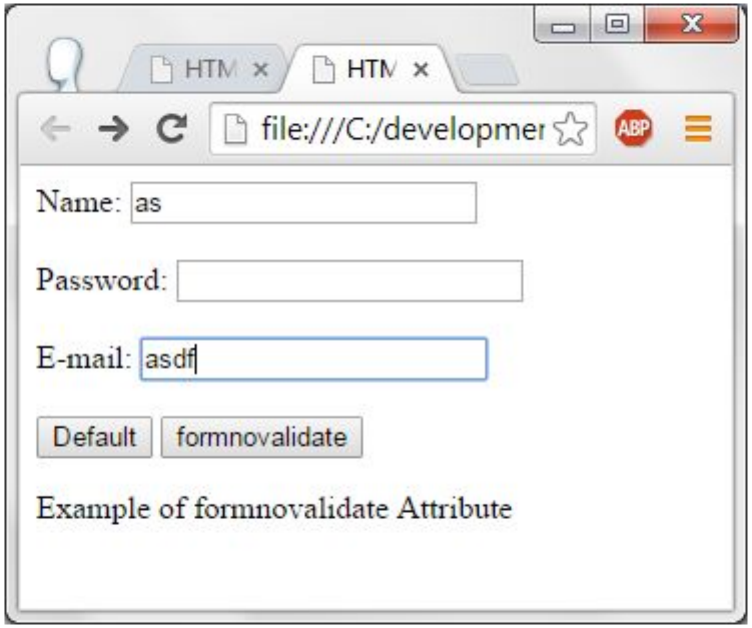
Example of multiple attribute

- With the multiple option in the input type, user can select multiple images from the image selection box.
- Click on choose files, image selection window will be displayed. User can select the images from this window.

Form attribute – novalidate.

```
<html>   <body>

     <form action="ab.php" method="post">

        <label for="name">Name:</label>

        <input type="text" name="name" id="name"
      class="textbox" /><br/><br/>

        <label for="mail">Password:</label>

        <input type="password" name="pwd" id="pwd"
      class="textbox" /><br/><br/>

        <label for="mail">E-mail:</label>

         <input type="email" id="mail" name="mail"
      class="textbox"/><br/><br/>

        <input type="submit" value="Default" class="btn"
      />

<input type="submit" value="formnovalidate" class="btn"
      formnovalidate />

       <p>Example of formnovalidate Attribute</p>
      </form>

   </body></html>
```

- Form with the fields and two buttons are displayed. When the first button is clicked form is validated and when the second button is clicked form is not validated.



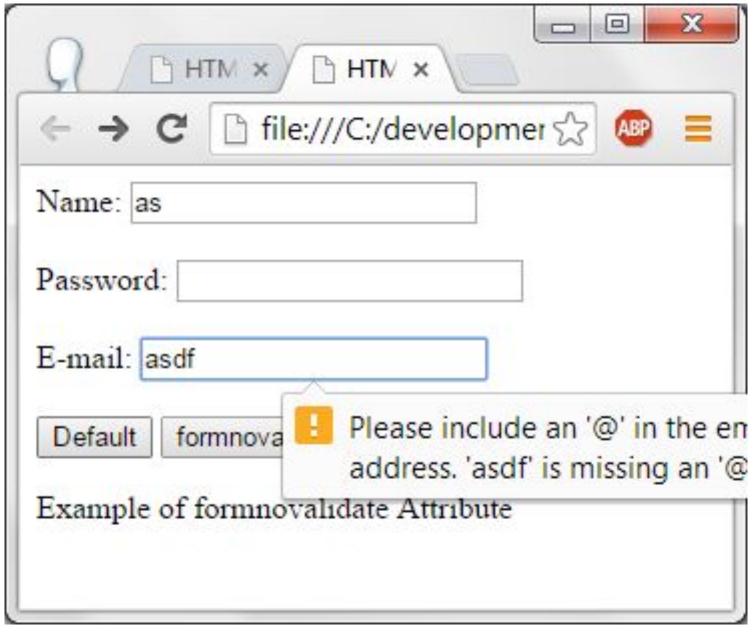
Name: as

Password:

E-mail: asdf

Default formnovalidate

Example of formnovalidate Attribute



Name: as

Password:

E-mail: asdf

Default formnova

Please include an '@' in the en
address. 'asdf' is missing an '@

Example of formnovalidate Attribute

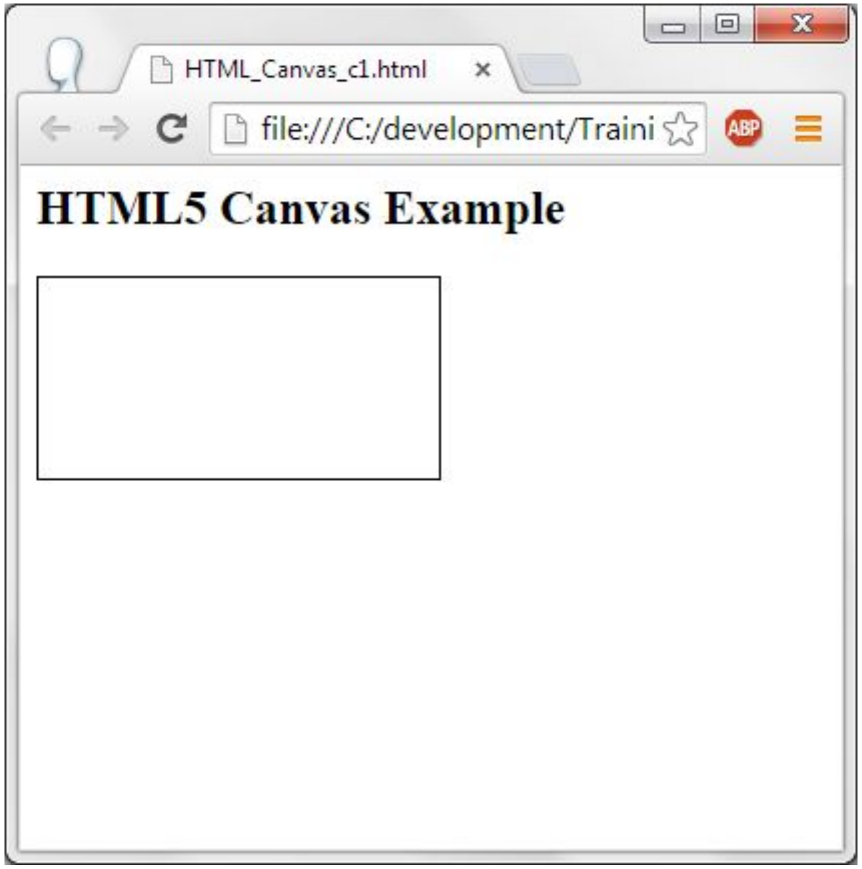
Canvas.

- Canvas element is used to draw graphics.
- Canvas uses script (Java script) to draw the graphics on the fly on web page.
- Canvas tag/element is a container. Script will be written to write graphics in this container.
- Boxes, text, images, lines can be written in Canvas container.

Canvas – container.

```
<html>

<head>

</head>

<body>

<h2>HTML5 Canvas Example</h2>

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

</body>

</html>
```

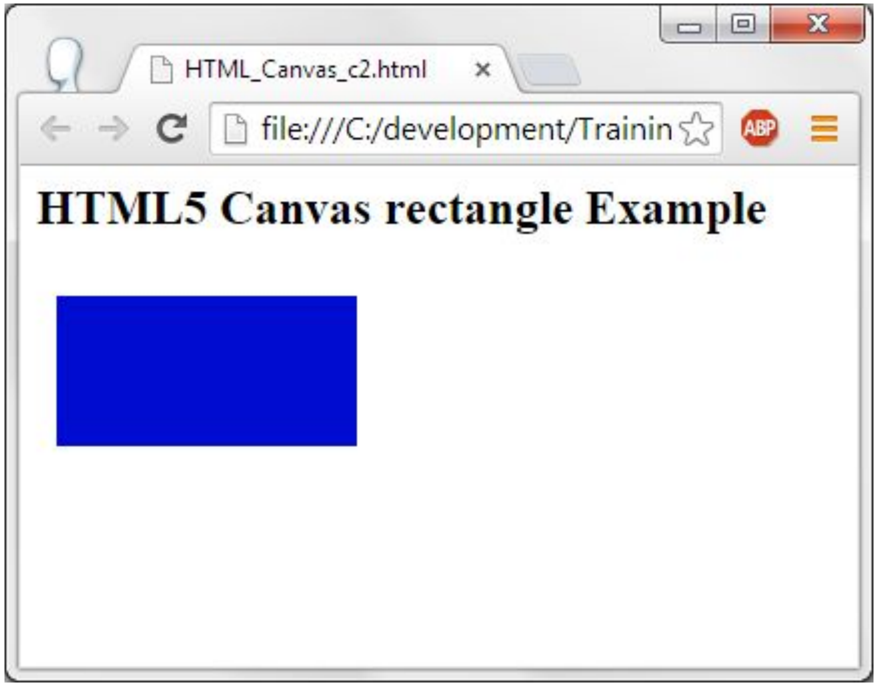
- In this code, canvas has the id “Canvas” with a width of 200 pixels and height of 100 pixels and border size is of 1px.



Canvas – Circle.

```
<html><head>

</head>

<body>

<h2>HTML5 Canvas rectangle Example</h2>

<canvas id="Canvas" width="200" height="100"
        style="border:1px;"></canvas>

<script>

var Canvas1 = document.getElementById("Canvas");

var context = Canvas1.getContext("2d");

context.fillStyle = "#000dd0";

context.fillRect(10,10,150,75);

</script></body></html>
```

- In the script, object of Canvas is obtained by getElementById().
- Obtained the context of the object.
- Fill style is specified and the rectangle co-ordinates are specified (x1, y1, width, height)



HTML5 Canvas rectangle Example

Canvas – Line

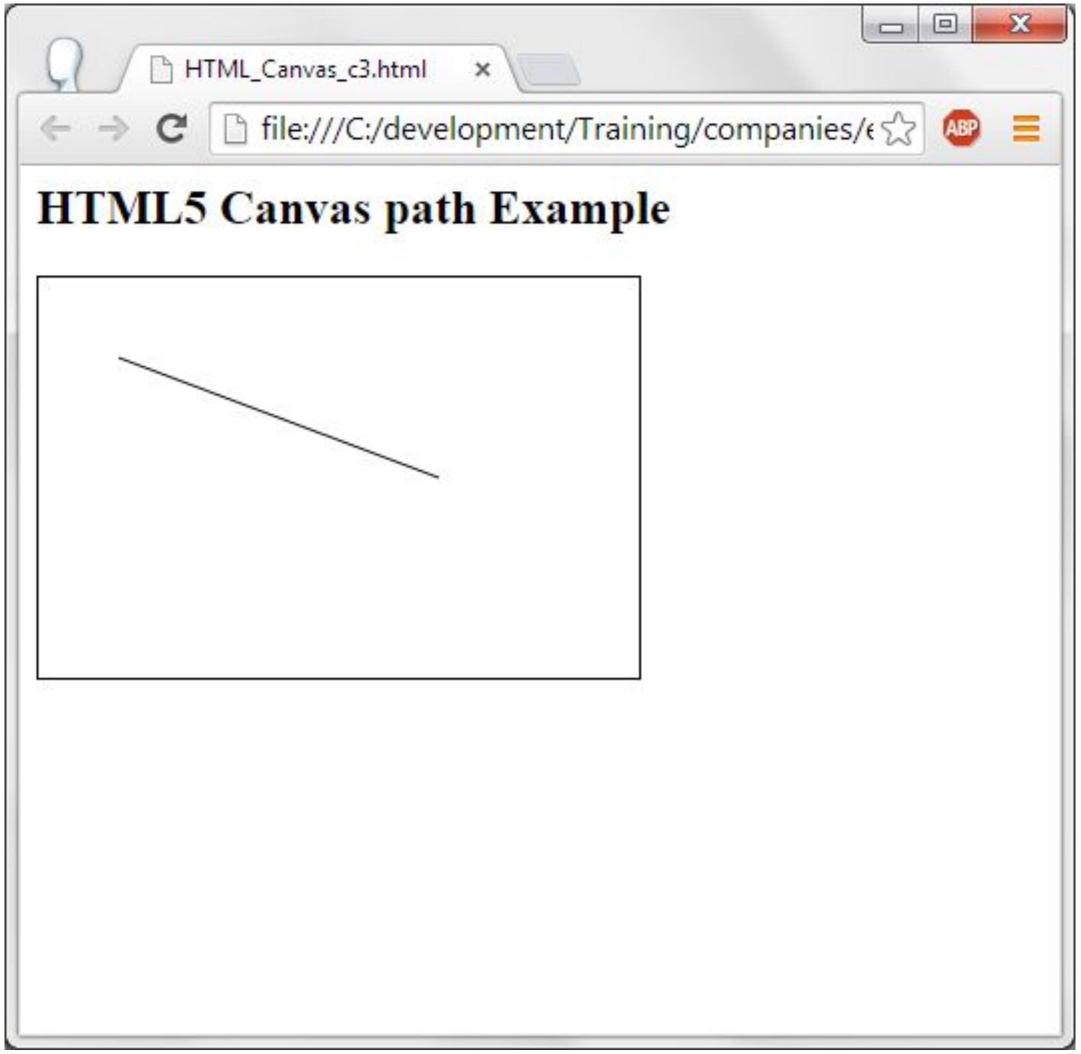
```
<html>

<head>

</head><body>

<h2>HTML5 Canvas path Example</h2>

<canvas id="Canvas1" width="300" height="200"
        style="border:1px solid #000000;"></canvas>

<script>

var Canvas1 = document.getElementById("Canvas1");

var context = Canvas1.getContext("2d");
```

context.moveTo(40,40);

context.lineTo(200,100);

```
context.stroke();</script></body>

</html>
```

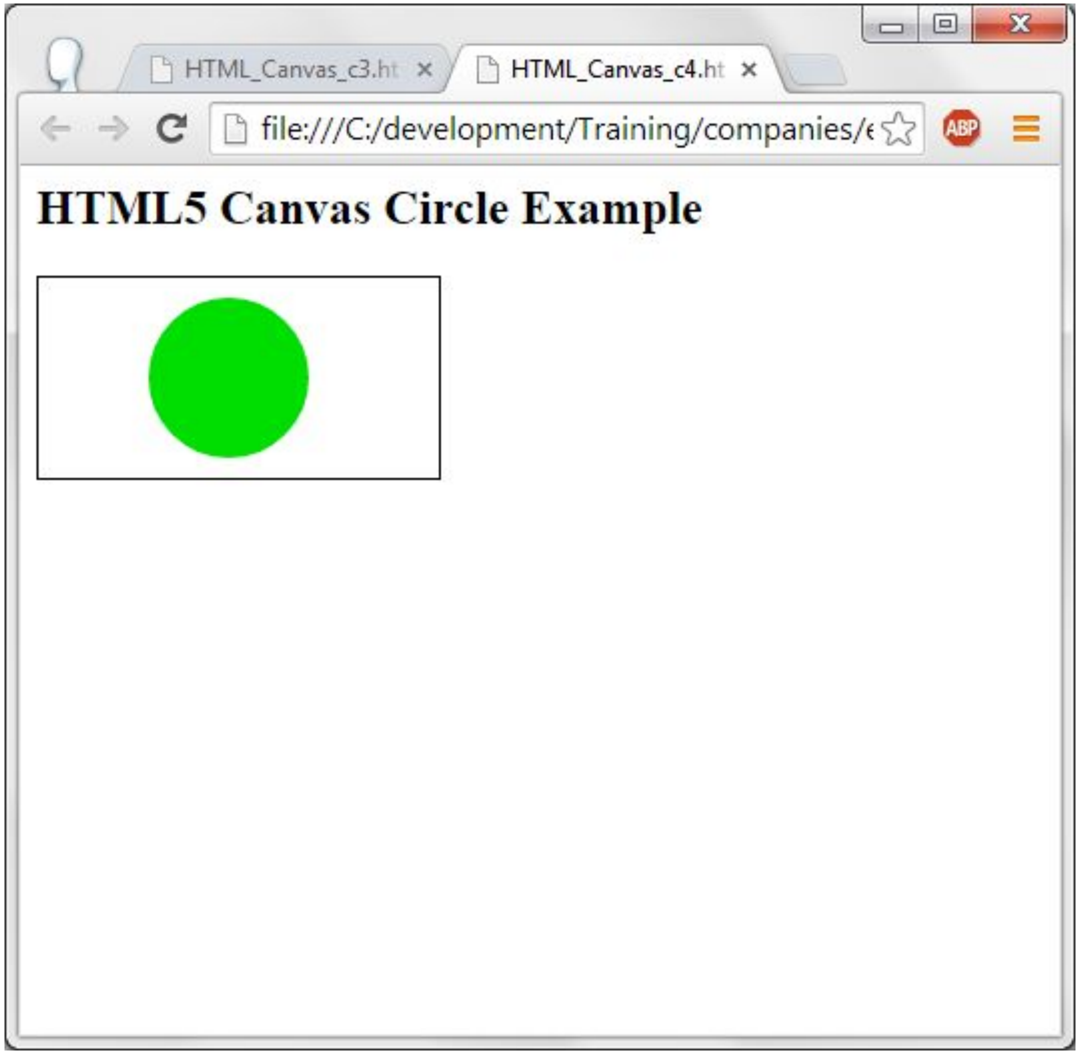
- In the canvas container, control is moved to 40,40 and the line is drawn till 200, 100



HTML5 Canvas path Example

Canvas - Arc

```
<html><head>

</head><body>

<h2>HTML5 Canvas Circle Example</h2>

<canvas id="Canvas1" width="200" height="100"
        style="border:1px solid #000000;"></canvas>

<script>

var Canvas1 = document.getElementById("Canvas1");

var context1 = Canvas1.getContext("2d");

context1.beginPath();

context1.arc(95,50,40,0,2*Math.PI);

context1.fillStyle = "#00dd00";

context1.fill();</script>

</body></html>
```

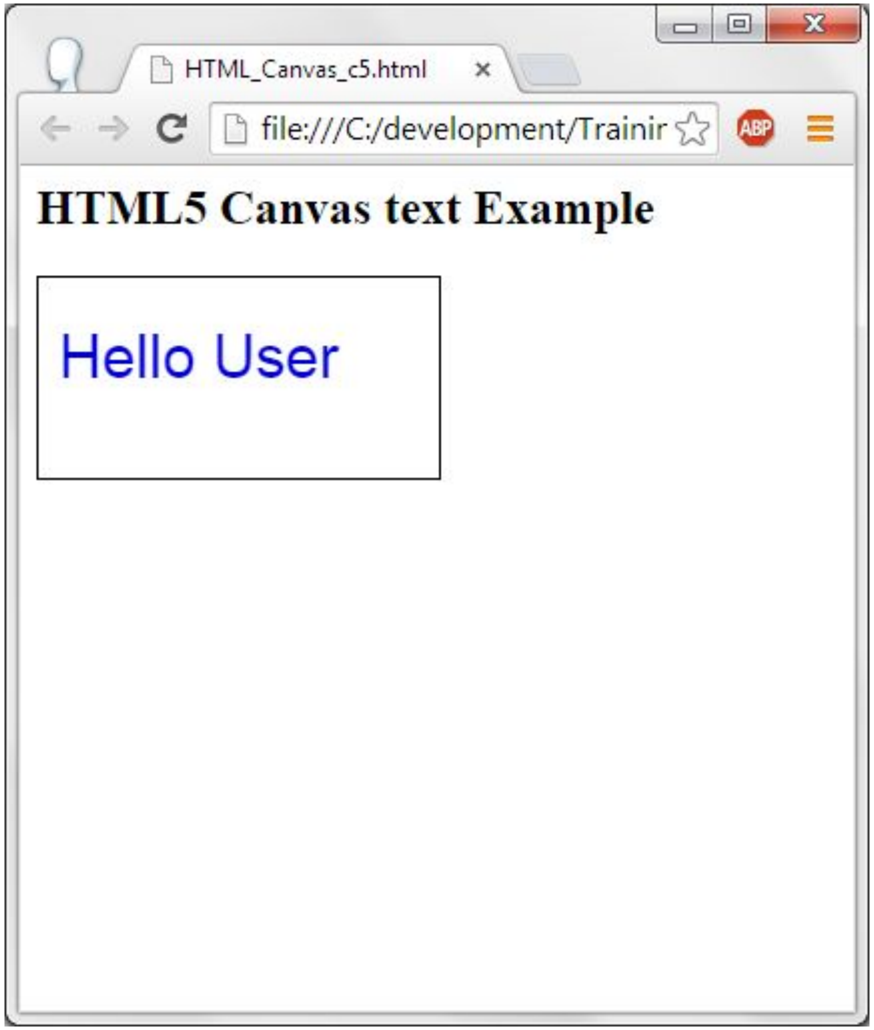


HTML5 Canvas Circle Example

- Syntax of arc is:
- Arc (x, y, radius, starting angle, ending angle)
- X and Y are the center point of the circle,
- Starting and ending angles are in radians hence 0 and 6.28 is given

Canvas - Text

```
<html><body>

<h2>HTML5 Canvas text Example</h2>

<canvas id="Canvas2" width="200" height="100" style="border:1px
        solid #000000;"></canvas>

<script>

var Canvas2 = document.getElementById("Canvas2");

var context1 = Canvas2.getContext("2d");

context1.fillStyle = "#0000dd";

context1.font = "30px Arial";

context1.fillText("Hello User",10,50);

</script></body></html>
```



HTML5 Canvas text Example

Hello User

- Filltext() is used to write text in Canvas container.

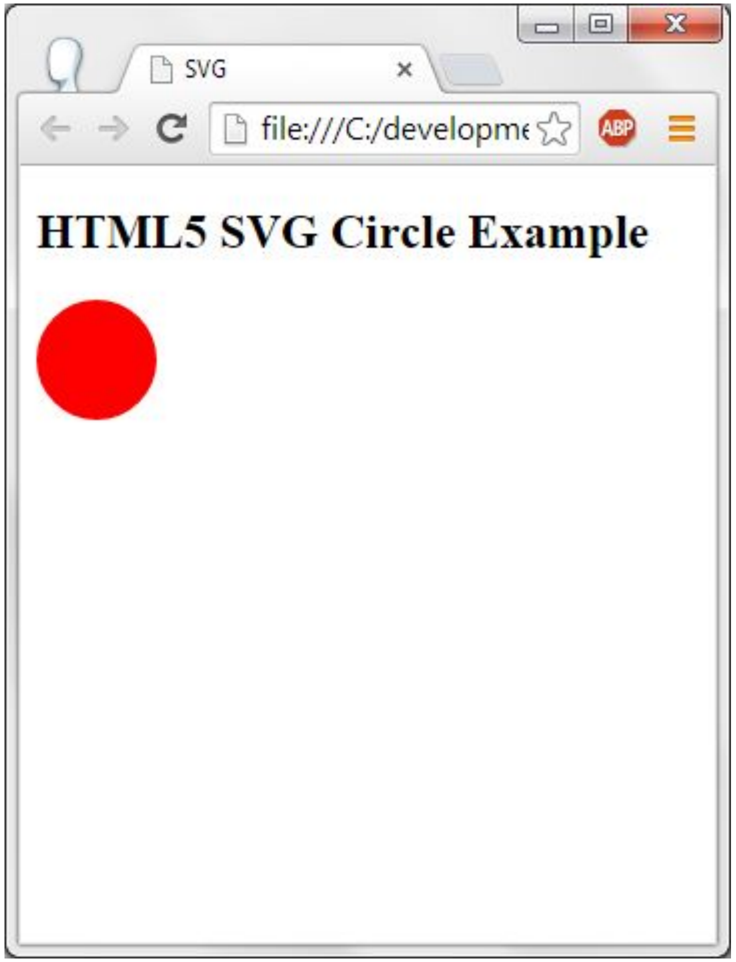
SVG

- SVG is used to write graphics on web page.
- SVG Stands for Scalar Vector Graphics.
- SVG does not lose the quality if they are zoomed.
- SVG can be printed in very high quality at any resolution.
- SVG is w3c Recommendation.
- SVG defines the graphics in XML format.

SVG – Circle.

```
<!DOCTYPE html>

<head>

<title>SVG</title>

<meta charset="utf-8" />

</head>

<body>

<h2>HTML5 SVG Circle Example</h2>

<svg id="ex1" height="100">

   <circle id="redcircle" cx="30" cy="30" r="30" fill="red" />


</svg></body></html>
```

- SVG is used to draw a circle by giving x, y co-ordinates and radius. Circle is drawn and fills with the color red.

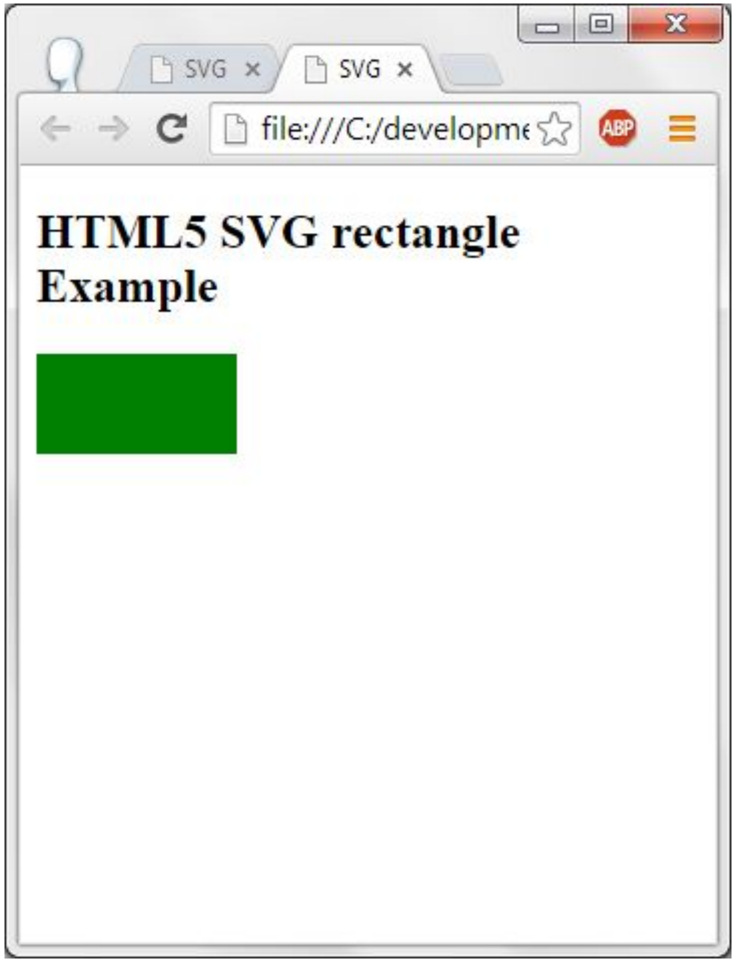


HTML5 SVG Circle Example

SVG – Rectangle.

```
<!DOCTYPE html>

<head>

<title>SVG</title>

<meta charset="utf-8" />

</head>

<body>

<h2>HTML5 SVG rectangle Example</h2>

<svg id="ex2" height="100" >

   <rect id="redrect" width="100" height="50" fill="green" />

</svg>

</body>

</html>
```

- SVG rectangle is drawn with width of 100 pixels and height of 50 pixels. After drawing the rectangle, it is filled with green color.

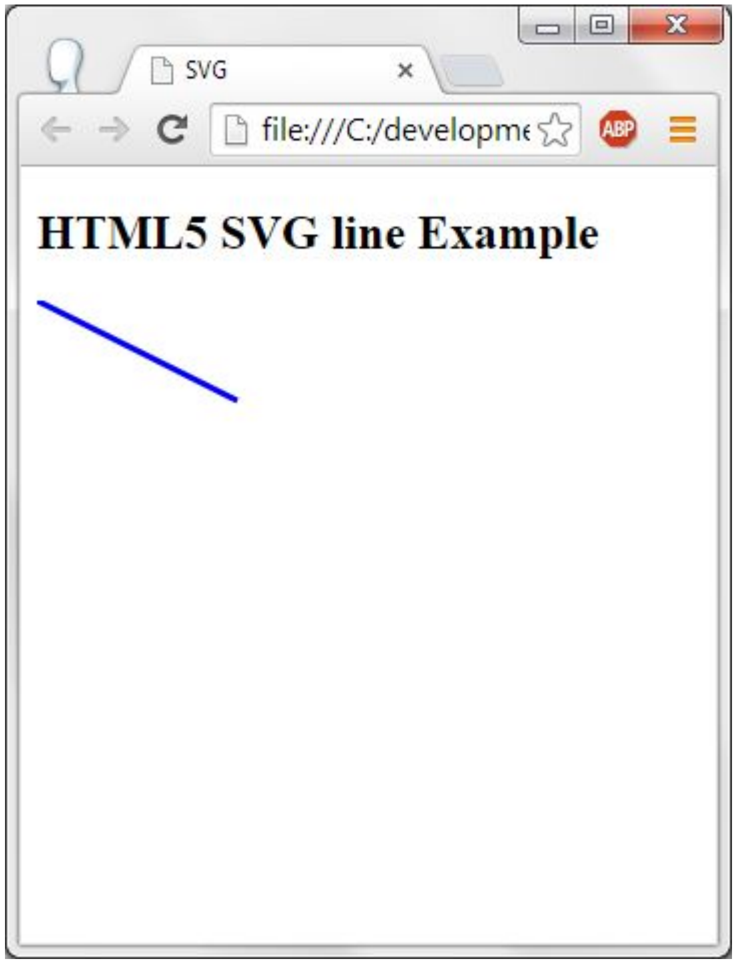


HTML5 SVG rectangle Example

SVG - Line

```
<!DOCTYPE html>

<head>

<title>SVG</title>

<meta charset="utf-8" />

</head><body>

<h2>HTML5 SVG line Example</h2>

<svg id="ex3 height="100" >

   <line x1="0" y1="0" x2="100" y2="50"

        style="stroke:blue;stroke-width:3"/>

</svg></body></html>
```

- X1 and y1 is the initial co-ordinates.
- Line is drawn from 0,0 to 100, 50. Line is drawn in blue color, width of 3 in size.

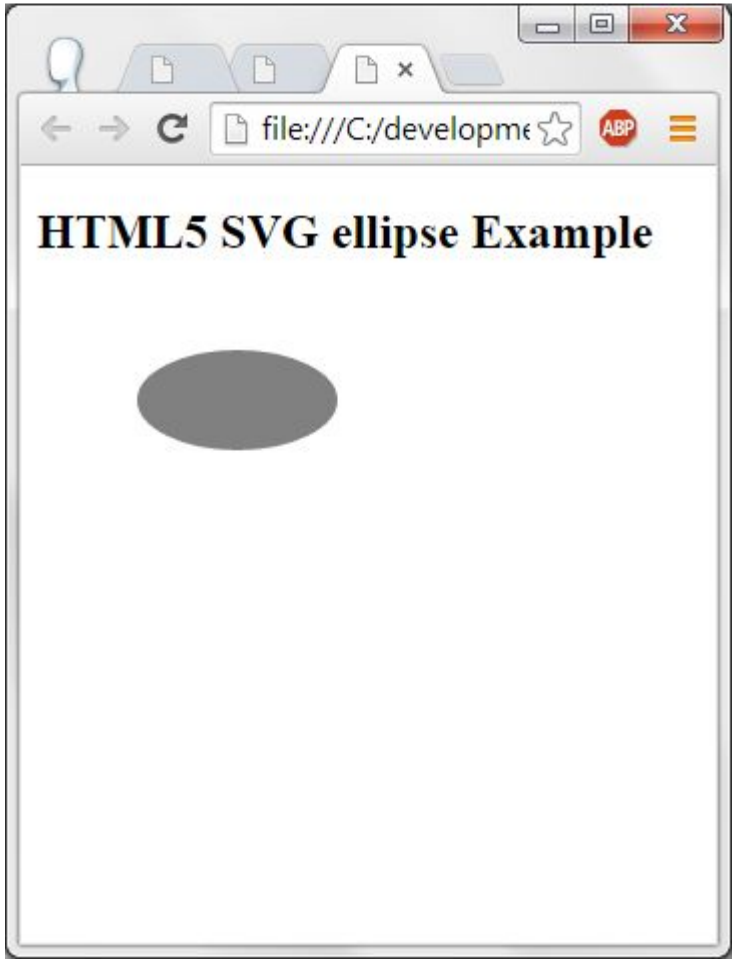


HTML5 SVG line Example

SVG - Eclipse

```
<!DOCTYPE html><head>

<title>SVG</title>

<meta charset="utf-8" />

</head>

<body>

<h2>HTML5 SVG ellipse Example</h2>

<svg id="ex4" height="170">

   <ellipse cx="100" cy="50" rx="50" ry="25" fill="gray"
      />

</svg></body></html>
```

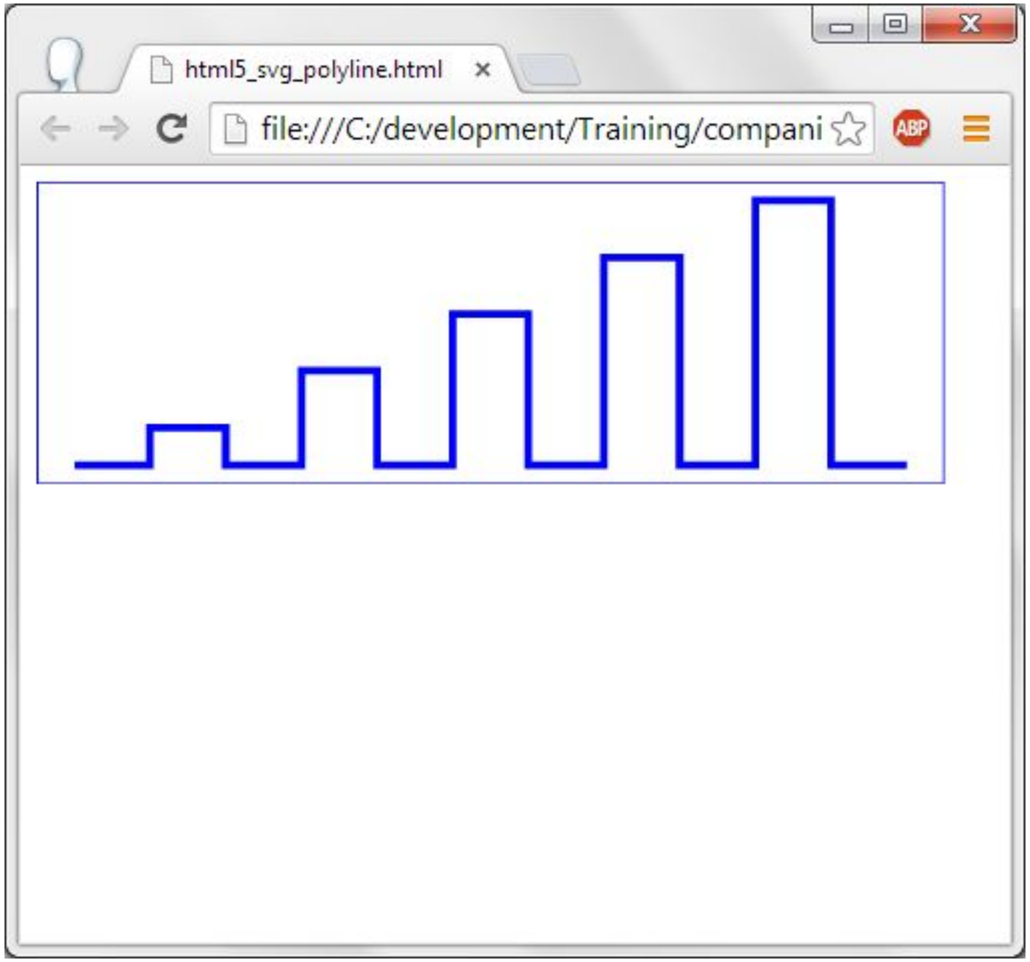
- At the location 100,50 an eclipse is drawn with 50 as x radius and 25 as the y radius and filled with gray in color.



HTML5 SVG ellipse Example

SVG-Polyline

```
<svg width="12cm" height="4cm" viewBox="0 0 1200 400"
    xmlns="http://www.w3.org/2000/svg" version="1.1">
  <desc>Example polyline01 - increasingly larger bars</desc>
  <!-- Show outline of canvas using 'rect' element -->
  <rect x="1" y="1" width="1198" height="398"
        fill="none" stroke="blue" stroke-width="2" />
  <polyline fill="none" stroke="blue" stroke-width="10"
            points="50,375
                    150,375 150,325 250,325 250,375
                    350,375 350,250 450,250 450,375
                    550,375 550,175 650,175 650,375
                    750,375 750,100 850,100 850,375
                    950,375 950,25 1050,25 1050,375
                    1150,375" />
</svg>
```



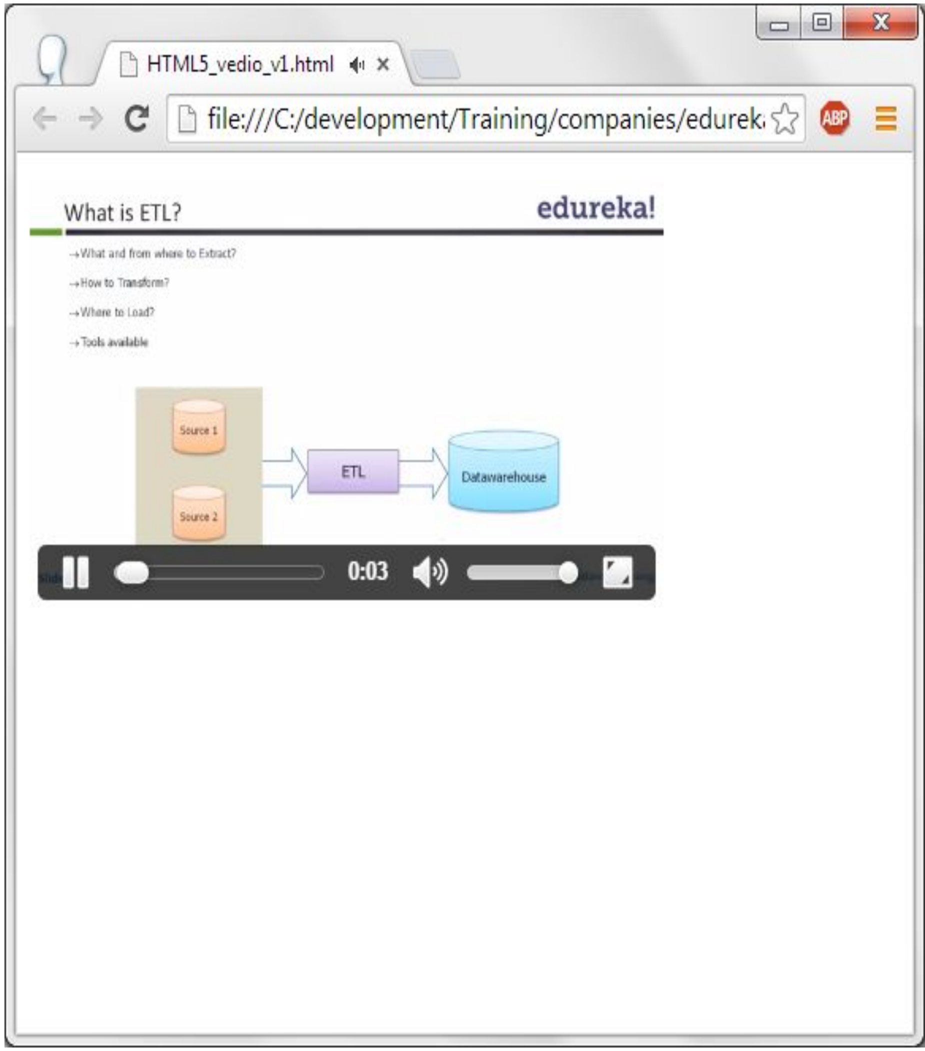
SVG-Polyline

Poly line syntax is – First two co-ordinates is the position control moves to. From there on a line is drawn for the next two (x,y) co-ordinates and so on.

```
<!DOCTYPE html>

<html><body>

<video id="vid" width="400" height="240" controls>

  <source src="vedio.mp4" type="video/mp4">

 </video>

</body></html>
```

HTML5 - Video

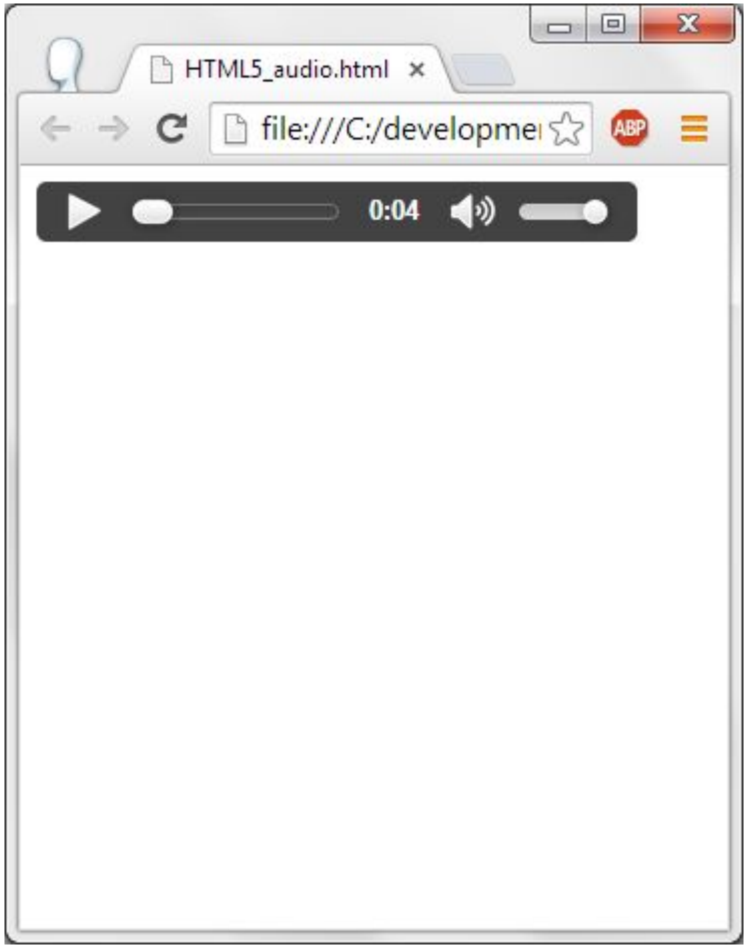
- Video tag is used to display the given video with the given height and width.
- Controls attribute is used to display the play/pause buttons and other controls on the video.
- Source of the video has to be specified using the source tag.
- Type of the video is specified as video/mp4.



HTML- Audio.

```
<!DOCTYPE html>

<html>

<body>

<audio controls>

  <source src="audio.mp3" type="audio/mpeg">

</audio>

</body>

</html>
```

- Audio tag is used to play the audio file. Here we are playing mp3 file.
- Controls will give the controls to play the audio and volume control is also given.
- File to be played is specified in the source tag. Type of the audio is specified as audio/mpeg.

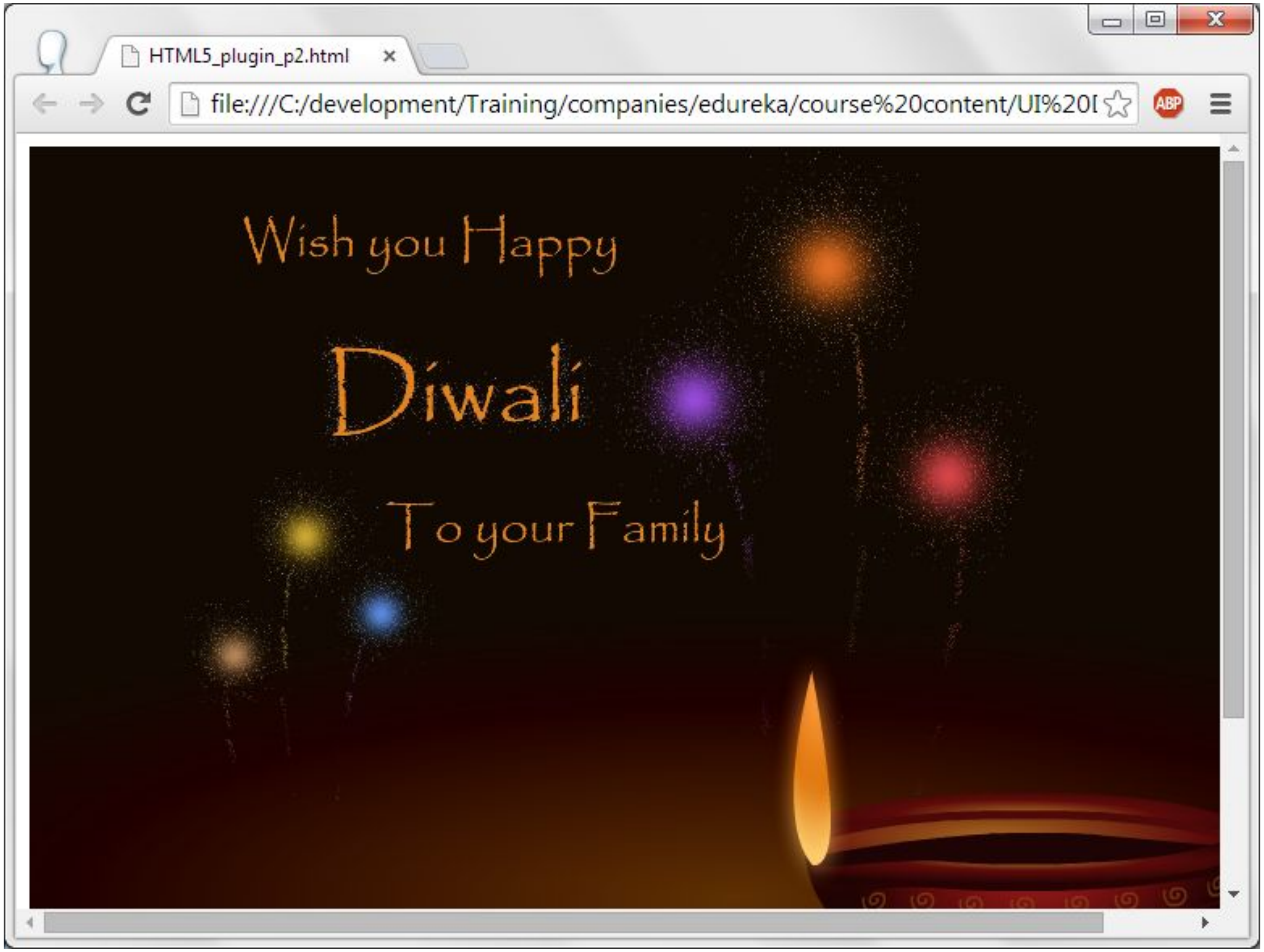


HTML5 – plugins.

- Tags <object> or <embed> are used to embed an object into the current web page. Code snippet is given below:
-
```
<html>
<body>
<object width="800" height="600" data="a.swf"></object>
</body>
</html>
```

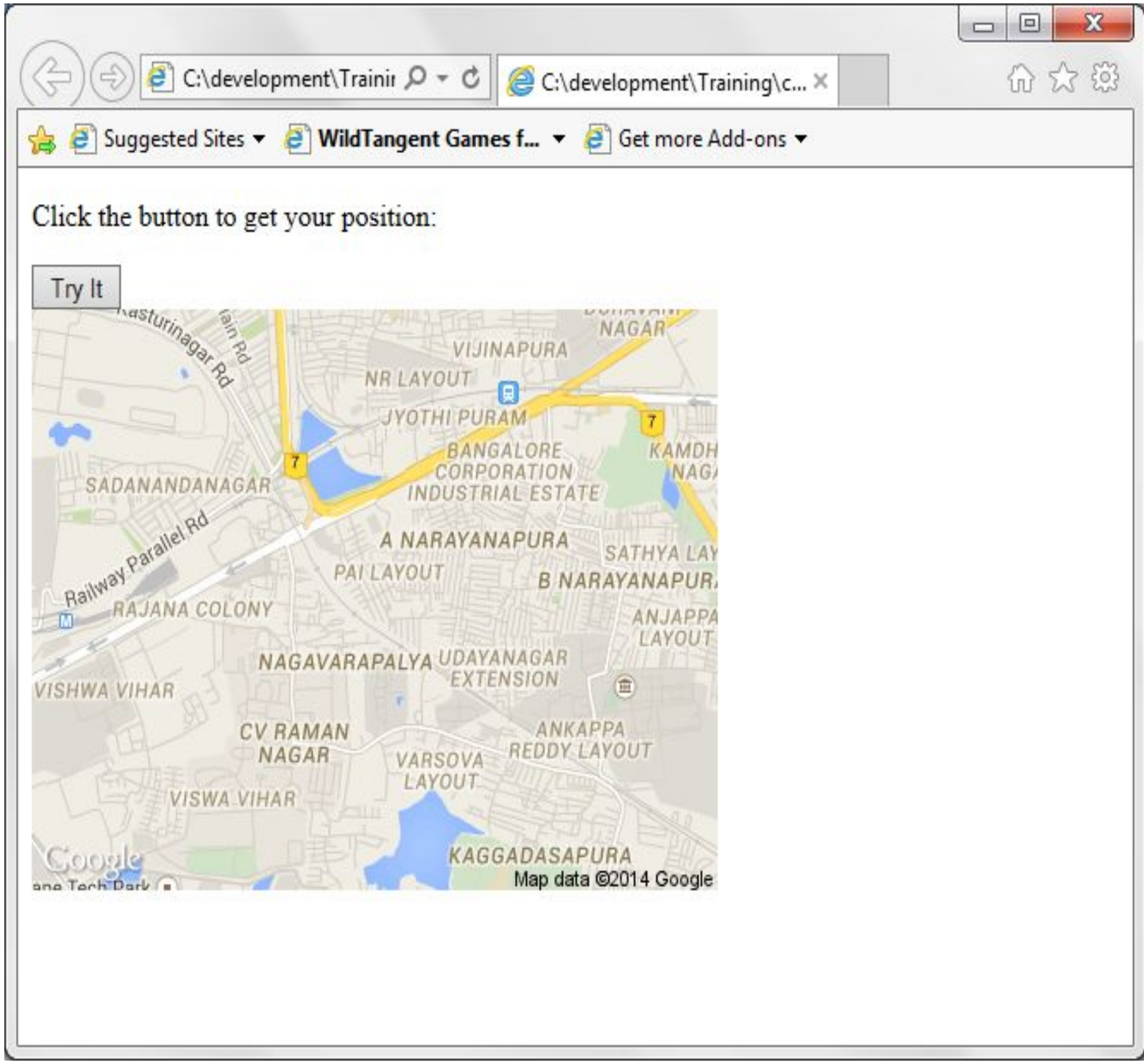
- Here a.swf file is embedded in the current web page.

HTML5 – plugins - Output.


HTML5_plugin_p2.html
file:///C:/development/Training/companies/edureka/course%20content/UI%20I
Wish you Happy
Diwali
To your Family

HTML5 – GEO Location.

```
<p id="demo">Click the button to get your position:</p>

<button onclick="getLocation()">Try It</button>

<div id="mapholder"></div>

<script>

var x = document.getElementById("demo");

function getLocation() {

   if (navigator.geolocation) {

      navigator.geolocation.getCurrentPosition(success,
      Error);

   } else {

      x.innerHTML = "Geolocation is not supported by this
      browser.";

   }}

function success(position) {    var latlon =
      position.coords.latitude+","+position.coords.longitude;

   var img_url =
      "http://maps.googleapis.com/maps/api/staticmap?cent
      er="

   +latlon+"&zoom=14&size=800x600&sensor=false";

   document.getElementById("mapholder").innerHTML =
      "<img src='"+img_url+"'>";}
```



Click the button to get your position:

HTML5 – GEO Location.

- On click on the button getLocaion() method is called.
- In this method, if the browser supports GeoLocation then current location is obtained by getCurrentPosition(). On sucess, it calls success() method. In this method, an image is created with the longtitude and latitude co-ordinates and this image is displayed. On any error, Error() method is called to display the current issue in displaying Geo Location.

HTML5 – Drag and Drop.

```
<!DOCTYPE HTML>

<html>

<head>

<script>

function allowDrop(ev) {

    ev.preventDefault();}

function drag(ev) {

    ev.dataTransfer.setData("text/html", ev.target.id);}

function drop(ev) {

    ev.preventDefault();

    var data = ev.dataTransfer.getData("text/html");

    ev.target.appendChild(document.getElementById(data));}</script></head>

<body background="desktop.png">

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)" border="1px" width="120" height="120">

<img id="drag2" src="recyclebin.png" width="100" height="100"/></div>

<img id="drag1" src="folder.png" draggable="true"

ondragstart="drag(event)" width="100" height="100">

</body>

</html>
```

HTML5 – Drag and Drop.

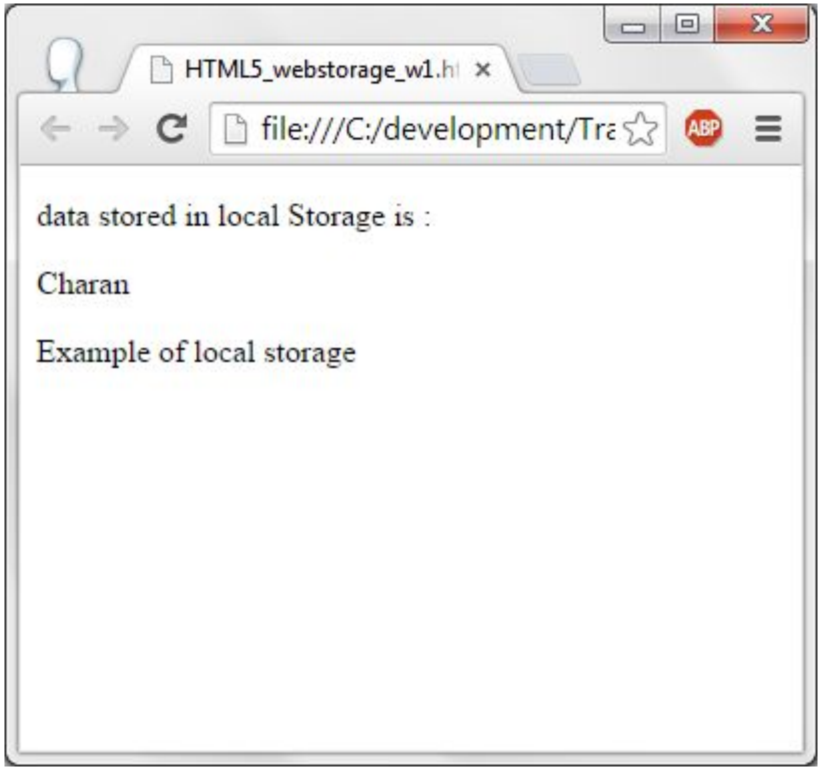
- Two images recycle bin and folder are displayed.
- Folder can be dragged and dropped into recycle bin.
- For the folder draggable=true is given so that this image can be dragged. When the folder image is started to drag, it calls a method drag (event). In this method, setting data type and id of the dragged image.
- You can allow folder to drag and drop on recycle bin making recycle bin by using onDragOver and OnDrop() methods. OnDragOver() method removes the default way of handling the elements by using the preventDefault() method. This is required to be used for drag and drop functionality. When folder is dropped on Recycle Bin, it calls OnDrop() which gets the data of the folder image and gets appended to recycle bin element.

HTML5 – Web Storage.

- local Storage is used to store the data locally about the client.
- Local Storage is stored on the current browser.
- SessionStorage is same as localStorage except that sessionStorage data gets deleted immediately after the browser is closed.
- It is better than cookies in terms of storage space.
- This data is never transferred to the server.

HTML5 – Web Storage.

```
<!DOCTYPE html>

<html>

<body>

<p> data stored in local Storage is : </p>

<div id="data"></div>

<script>

// Check browser support

if (typeof(Storage) != "undefined") {

   // Store

   localStorage.setItem("Username", "Charan");

   // Retrieve

   document.getElementById("data").innerHTML =
      localStorage.getItem("Username");

} else {

   document.getElementById("data").innerHTML = "No Browser
      support";}

</script>

<p>Example of local storage</p>

</body></html>
```



data stored in local Storage is :

Charan

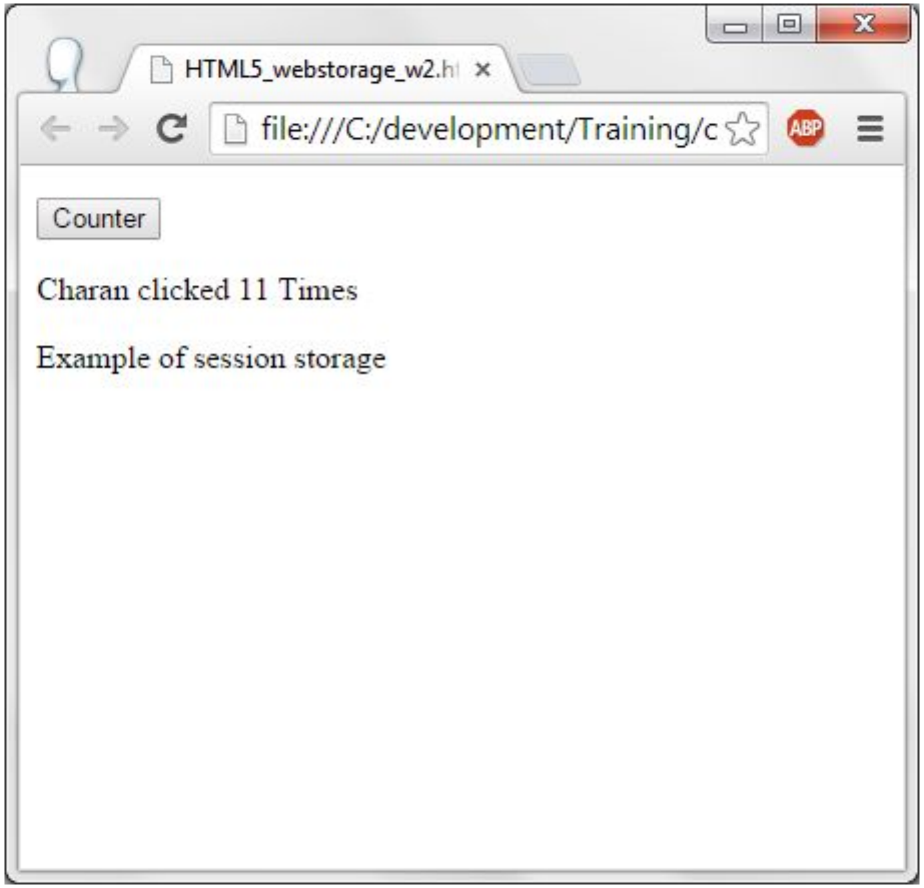
Example of local storage

HTML5 – Web Storage.

- Checks whether browser support for web storage by using typeof(Storage) != "undefined". If the browser supports then **charan** is set as Username in localStorage.
- Id “**data”** value is set by obtaining the value of Username variable from localStorage. This is done by using **document.getElementById("data").innerHTML = localStorage.getItem("Username");**

HTML5 – session Storage.

```
<!DOCTYPE html>

<html><head><script>

function Counter() {

   if(typeof(Storage) !== "undefined") {

      if (sessionStorage.Counter) {

         sessionStorage.Counter = Number(sessionStorage.Counter)+1;

      } else {

         sessionStorage.Counter = 1;       }

if(!sessionStorage.user){

sessionStorage.user = "charan";}

      document.getElementById("data").innerHTML
      =sessionStorage.user +" clicked "+ sessionStorage.Counter ;

   } else {

      document.getElementById("data").innerHTML = "No browser
      support";    }}

</script></head>

<body>

<p><button onclick="Counter()" type="button">Counter</button></p>

<div id="data"></div>

<p>Example of session storage</p>

</body></html>
```



Charan clicked 11 Times

Example of session storage

HTML5 – session Storage.

- If the counter is not defined in sessionStorage then counter value is set to 1 else value the counter is read and its value is incremented by one unit and stored in sessionStorage counter.
- If the user name is not there in Charan will be set as user in sessionStorage.
- Click count is displayed on the element data by obtaining user and Count value from sessionStorage.
- Note : Once the browser is closed then these values are lost.

HTML5 – Application cache.

- It is possible to cache a web site. Advantage of having application cache is, even when internet is not connected the cached web site can be displayed.
- It has 3 advantages:
  - A web page can be accessed without internet connection.
  - Cached pages are always faster.
  - It reduces the load on the server as it does not reach the server to display the web page.
- To use the cached web page, **manifest** attribute has to be specified in <html> tag as given below:
  - <html manifest=”mypage.appcache”>

  mypage will have the files to be loaded when the net is not connected.
- Note that the html file which has manifest will only be cached otherwise it is not cached.

HTML5 – Web worker.

- A web worker is a task running in the background which is written in java script without affecting the performance of the page being displayed.
- The main reason is, when a java script is invoked, till it is finished page is not active. To resolve this issue, when it is possible to run the task in the back ground then web worker can be used.
- When the js file is running, any task like selecting, clicking on web page can be continued as usual.

HTML5 – Web worker.

```
<!DOCTYPE HTML>

<html>

<head>

<title>Big for loop</title>

  <script>

      function sayHello(){

         alert("Hello sir...." );      }

  </script></head>

<body>

<div id="data">Here the data will be displayed</div>

   <input type="button" onclick="startWorker();" value="Big Loop" />

   <input type="button" onclick="sayHello();" value="Say Hello" />

    <input type="button" onclick="stopWorker();" value="Stop loop" />

</body>
```

HTML5 – Web worker.

```
<script>

var wrkr;


function startWorker() {

   if(typeof(Worker) !== "undefined") {

      if(typeof(wrkr) == "undefined") {

         wrkr = new Worker("internalfile.js");

      }

      wrkr.onmessage = function(event) {

         document.getElementById("data").innerHTML = event.data;

      };

   } else {

      document.getElementById("data").innerHTML = "No support for browser";

   }}

function stopWorker() {

   wrkr.terminate();

   wrkr = undefined;}

</script></html>
```

HTML5 – Web worker.

Internalfile.js

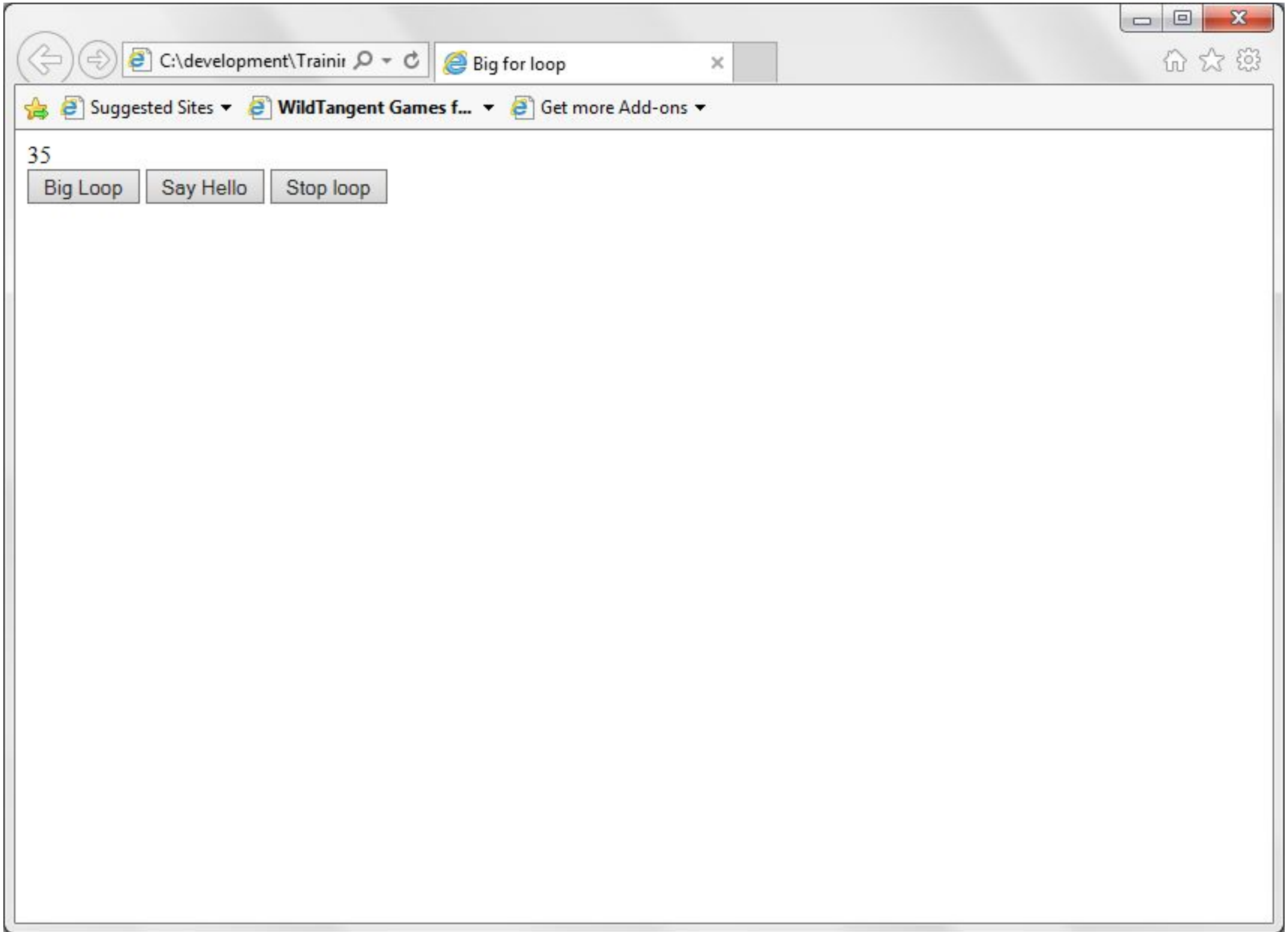
+++++++++

```
var i = 0;
function bigLoop() {
    i = i + 1;
    postMessage(i);
    setTimeout("bigLoop()",500);
}
bigLoop();
```

HTML5 – Web worker.

- 3 buttons are created. Big Loop, say hello and stop loop. When the big loop is clicked it prints the numbers from 1 and when stop loop is clicked it stop the displaying of further numbers. When say hello button is clicked then Hello Sir alert message will be displayed.
- On click of Big loop calls a method start worker. If the worker object is not defined and wrkr variable is not defined then new worker object is defined by specifying the internalfile.js file. This means internalfile.js is the js file which runs in the background.
- When ever there is a message from the web worker then a function is executed which displays the value which is given by the web worker (from js file).
- When the stop button is clicked then web worker is terminated.
- When displaying the numbers, you can click on say hello. Which displays the message.
- Internal.js file initializes the variable i to 0. has a function bigLoop(). In this function, value of i get incremented by one unit and this value is posted to the calling file which is the html file. SetTimeout calls the specified function in given milli seconds. Here we are calling bigLoop() every half second.

HTML5 – Web worker.



HTML5 - SSE

- SSE stands for Server Sent Events.
- This is when the server sends the updates to the browser or the client.
- Web page should be open to receive the events or messages from the server.
- Some of the examples of Sse are Facebook updates, Sports results, stock price updates etc.,
- Java script uses EventSource() method with the server side object which sends the messages or events.
- Each time a message is received from the server OnMessage event occurs.
- Java Script should handle the OnMessage event and perform the required operation.

HTML5 – SSE.

```
<!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>
```

HTML5 - SSE

- demo_sse.php is the server which will sends the messages.
- On receiving the message from this source a function is called. In this function, the data which is received from the server is displayed on the browser.

Thank you!