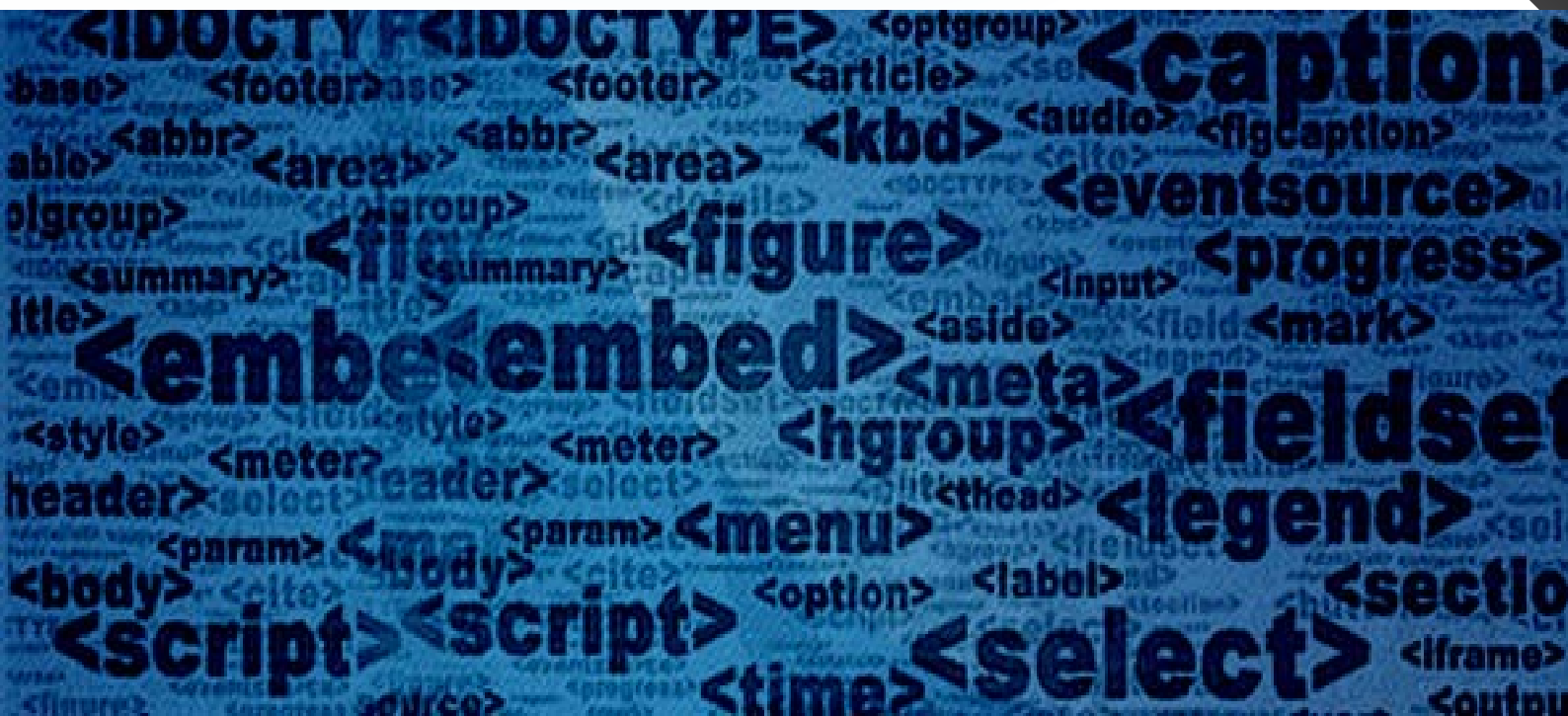
True Vision
Technologies

HTML5



INTRODUCTION

Title		Page No.
1	HTML 5 – Gaining interest to become a Trend	3
2	3D graphics , effects and HTML5	7
3	MULTIMEDIA AND HTML5	49
4	HTML 5 SEMANTICS and HTML5 FORMS	53

HTML 5 – Gaining interest to become a Trend

Today's designers when asked about HTML5 do hesitate to answer because of the lack of knowledge about HTML5.

Also many times a question arises in many people's mind that what's the difference between HTML and HTMl5. Many people think it's just the updated version of HTML. This is also just due to the lack of knowledge about HTML5.

Before discussing about HTML5, let's have a look at some of the wrong notions about it.

HTML 5 – Gaining interest to become a Trend

1 **Many people think why we shall use HTML5 if it isn't working on the old version browsers?**

Yes, that's true that HTML5 is not supported by older versions of browsers, but instead of using HTML5 for the complete website, you can at least use some features of it. You may not get support for HTML5 but cavn get support for some of it features.

2 **While thinking of HTML, the first thing that comes in your mind is a variety of tags and after that the '< >' brackets.**

That's a truth that HTML is made up of this tags and '< >' brackets only but HTML5 gives an in depth meaning of this '< >' brackets as well as tags. HTML5 defines the interaction of this '< >' brackets with the JavaScript through DOM (Document Object Model).

3 **People think that lets redesign our website with HTML5. But then due to lack of knowledge, they don't use it. But actually you don't need to do so. You can use that old HTML with HTML5.**

No one would disagree on this that HTML is the best markup language ever. And HTML5 is built on that HTML only with some more features. So it does support the features and will work well on the website developed on HTML.

4 **Is it very easy to get started with HTML5?**

Yes, it is very easy to get started with HTML5. HTML supported many different types of DOCTYPE but HTML5 just supports one i.e. <!DOCTYPE html>

Whether you want to draw on a canvas, play video, design better forms, or build web applications that work offline, you'll find that HTML5 is already well-supported. Firefox, Safari, Chrome, Opera, and mobile browsers already support canvas, video, geolocation, local storage, and more. Google already supports micro data annotations. Even Microsoft — rarely known for blazing the trail of standards support — supports most HTML5 features in Internet Explorer 9.

Now after discussing about the wrong notions let's discuss about what is HTML5 and why was it required to be developed.

HTML5 is a result of the co-operation between the W3C (World Wide Web Consortium) working on XHTML and WHATWG (Web Hypertext Application Technology Working Group) working on web forms and applications. Its core aim is to improve the language that supports multimedia easily without using the scripting.

HTML 5 – Gaining interest to become a Trend

Reasons why HTML5 was developed are:

- Reduce the requirement of external plugins.
- Better error handling.
- Such markup that replaces the difficult scripting. After reasons, let's come upon the new features that were introduced in HTML5 are:
- New Markup
- New API's
- Support for local storage
- Error Handling

Now let's look at each and every feature in detail. Let's start with New Markup.

- In HTML5, many tags deprecated in HTML are removed like <font>, , <dir>, <frame>, etc.
- Some many common blocks for which tag was to be used, now is developed using the tags named after the name of the block. Like for footer i.e. the bottom of each and every webpage can now be written in <footer> tag. And this tags are also called as Structural tags.
- For playing an audio or displaying a video, we used <object> tag but now we have individual tags for both playing an audio and displaying a video i.e. <audio> and <video> tag respectively.
- Some new form tags are also introduced like <datalist>, <keygen>, <output>, etc.
- The <canvas> tag for immediate mode 2D drawing.

After discussing on the new markups designed, let's discuss upon the New API's introduced in HTML5.

- Timed media playback
- Offline Web Applications
- Document editing
- Drag-and-drop
- Cross-document messaging
- Browser history management
- MIME type and protocol handler registration
- Micro data

HTML 5 – Gaining interest to become a Trend

- Web Storage, a key-value pair storage framework that provides behavior similar to cookies but with larger storage capacity and improved API.
- Scalable Vector Graphics
- GeoLocation - The most important API introduced.

So this was a nut shell of the whole HTML5 and its features but you wouldn't like it if I just listed the features. So I won't disappoint you. Here is the detail description about each and every markup designed or API introduced.

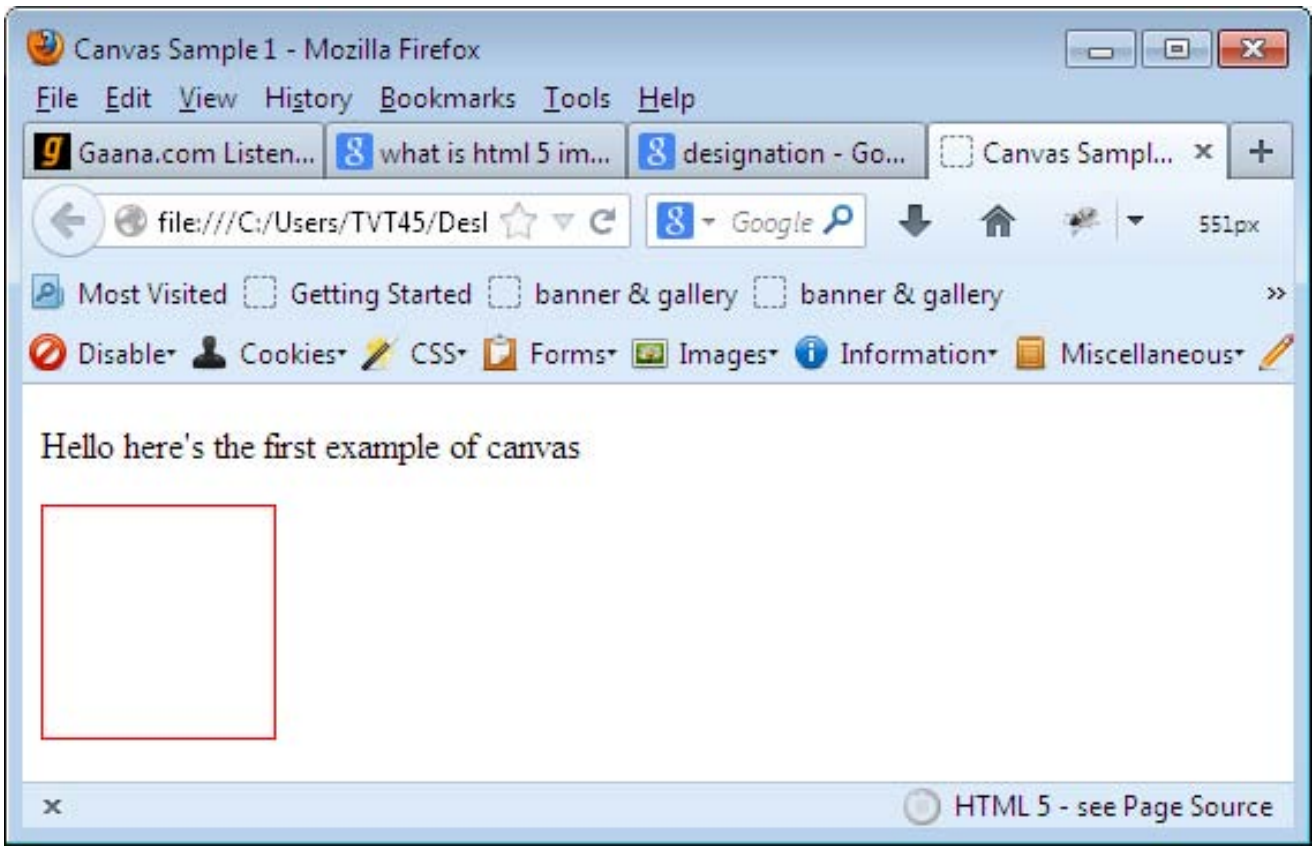
<canvas> tag

- The canvas tag is used to draw 2d objects. It's just draws the outline. For the actual graphics you need to use scripts with it.
- **Syntax:** <canvas> </canvas>

To create a simple canvas:

```
<!DOCTYPE html>
<html>
<head>
<title> Canvas Sample 1</title>
</head>
<body>
        <p>Hello here's the first example of canvas</p>
        <canvas width="100px" height="100px" style="border: 1px solid red ;"> Hello</canvas>
</body>
</html>
```

Output :



- Canvas with JavaScript is used to draw lines, circles, to write text, to show gradients, and to show images in the canvas.

<svg> element

- SVG stands for Scalable Vector Graphics.
- SVG is a vector graphics format based on XML means SVG is an XML language, similar to XHTML, which can be used to draw graphics.
- SVG images and their behaviors are defined in XML file. As XML, SVG files can be created and edited in text editors.
- SVG can be used to develop lines, shapes, animated objects, to write text, etc.
- It can be used to create an image either by specifying all the lines and shapes necessary, by modifying already existing raster images, or by a combination of both.
- The image and its components can also be transformed, composited together, or filtered to change its appearance completely.
- **SVG supports some predefined shapes in the form of elements. They are:**
  - Rectangle <rect>
  - Circle <circle>
  - Ellipse <ellipse>
  - Line <line>
  - Polyline <polyline>
  - Polygon <polygon>
  - Path <path>
- **Syntax:** <svg xlmns = "http://www.w3.org/2000/svg" version="1.1"> </svg>

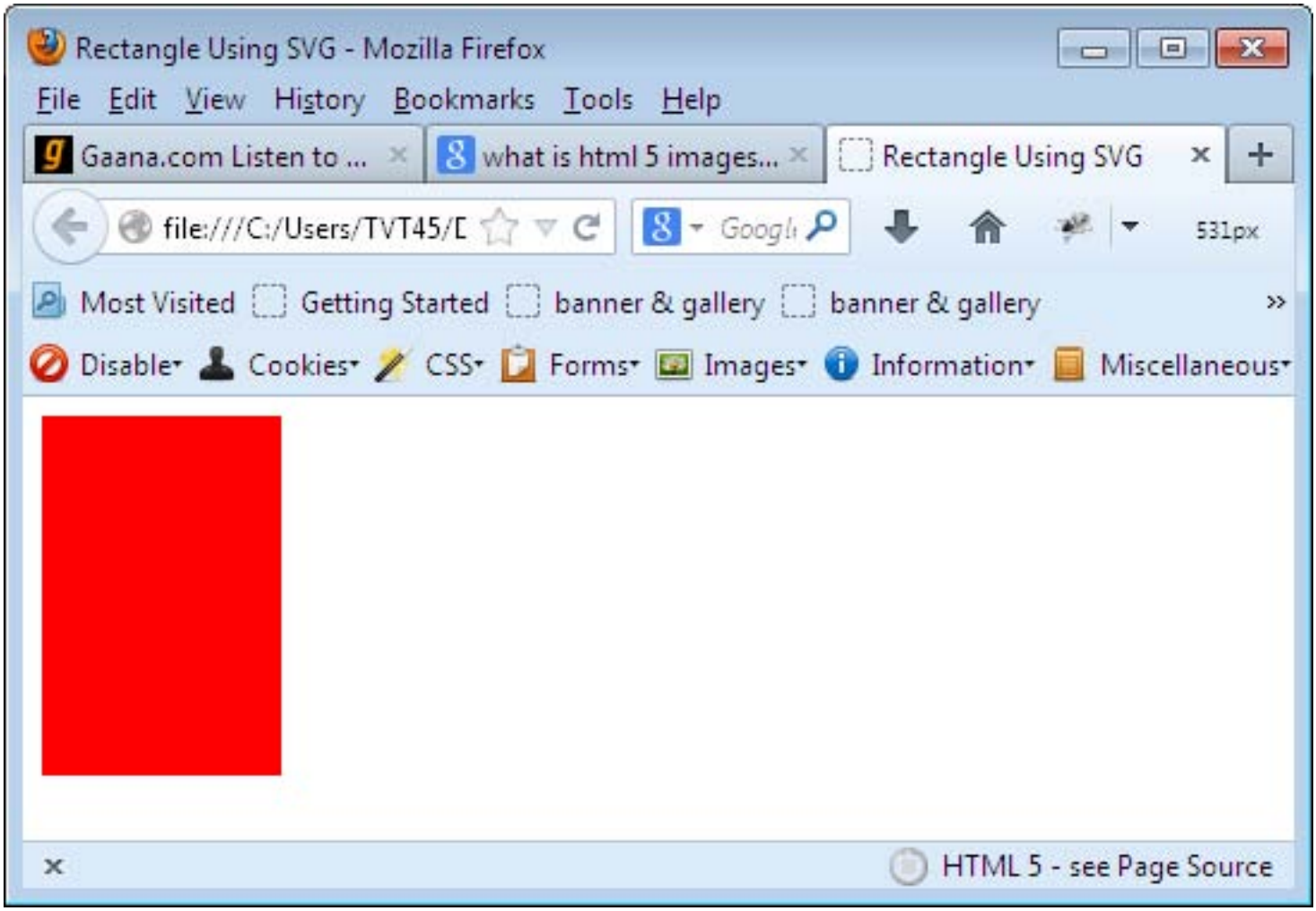
SVG RECTANGLE: <rect> tag

- To create a rectangle we use <rect > tag.

Example of a Simple Rectangle.

```
<!DOCTYPE html>
<html>
<head>
<title> Canvas Sample 1</title>
</head>
<body>
        <p>Hello here's the first example of canvas</p>
        <canvas width="100px" height="100px" style="border: 1px solid red ;"> Hello</canvas>
</body>
</html>
```

Output :



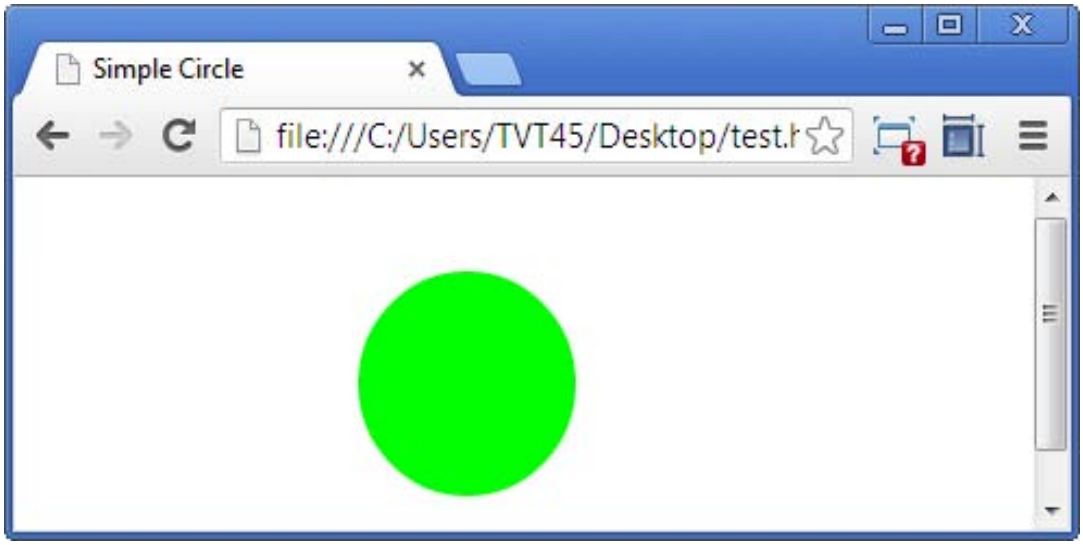
- **The attributes of the <rect> tag are:**
  - **width and height :** Defines the dimensions of the rectangle.
  - **CSS fill :** used to fill color in the rectangle.
  - **CSS stroke :** used to define the color of the border.
  - **CSS stroke – width:** used to define the width of the border.
  - **CSS fill – opacity:** used to provide opacity (transparency) to the color filled in the rectangle.
  - **CSS stroke – opacity:** used to provide opacity (transparency) to the color of the border in the rectangle.
  - **x and y:** defines the position of the rectangle on the web browser.

SVG CIRCLE: <circle> tag

- To create a circle we use <circle > tag.
- **Example of a Simple Circle.**

```
<!DOCTYPE html>
<html>
        <head>
                <title> Simple Circle </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">

                        <circle cx="200" cy="200" r="50" style="fill:rgb(0,255,0);" />
                </svg>
        </body>
</html>
```

Output :



- **The attributes that can be used with <circle> tag are:**
  - **cx and cy:** These attributes are defined to set the center of the circle.
  - **r:** It defines the radius of the circle.
  - **CSS fill:** used to fill color in the circle.
  - **CSS stroke:** Used to define the color of the border around the circle.
  - **CSS stroke-width:** Used to define the width of the border around the circle.

SVG ELLIPSE: <ellipse> tag

- To create an ellipse we use <ellipse > tag.
- Example of a Simple Ellipse.

```
<!DOCTYPE html>
<html>
        <head>
                <title> Simple Ellipse Example </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <ellipse cx="100" cy="100" rx="50" ry="75" />
                </svg>
        </body>
</html>
```

Output :

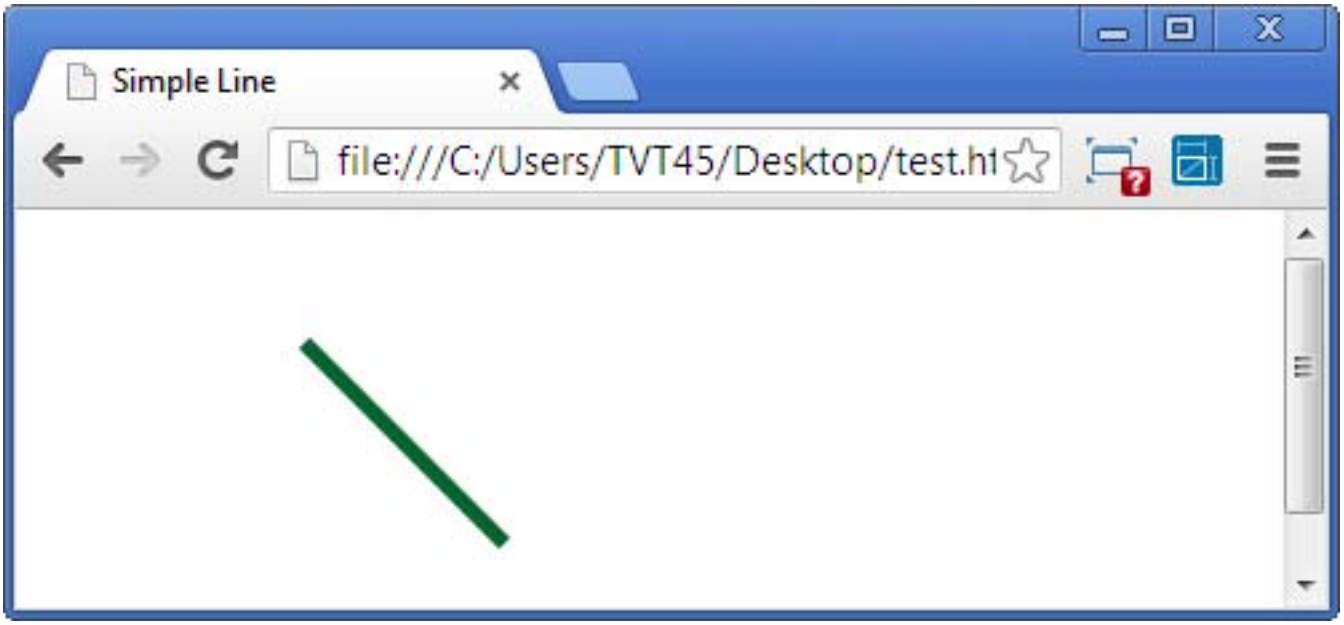
- **The attributes used with <ellipse> tag are:**
  - **cx and cy:** These attributes define the center of the ellipse.
  - **rx and ry:** These attributes define the radius of the ellipse in horizontal as well as vertical direction respectively.
  - **CSS fill:** Used to fill the color in ellipse. By default the color is black.
  - **CSS stroke:** Used to define the color of the border around the ellipse.
  - **CSS stroke – width:** Used to define the width of the border around the ellipse.

SVG LINE: <line> tag

- To create a line we use <line> tag.
- Example of a Simple Line.

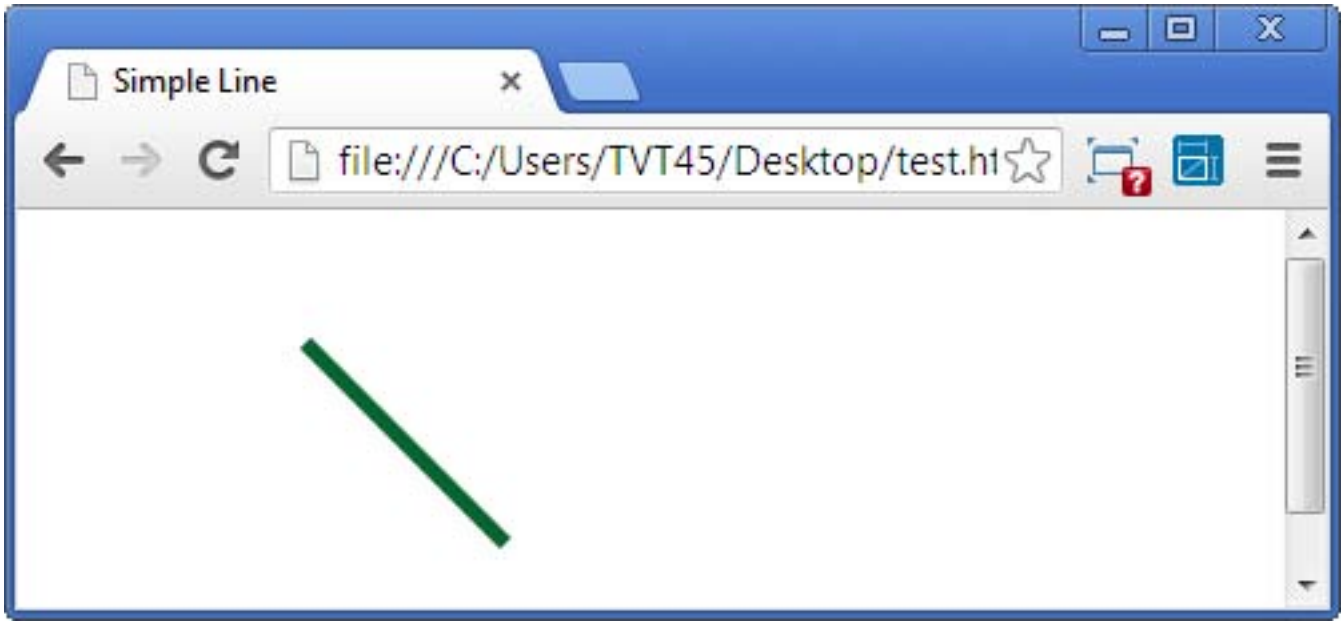
```
<!DOCTYPE html>
<html>
        <head>
                <title> Simple Line </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <line x1="100" y1="120" x2="175" y2="195" style="stroke:rgb(10,100,50);
stroke-width:6" />
                </svg>
        </body>
</html>
```

Output :



- **The attributes used with line tag are:**
  - **x1 and y1:** They define the initial or the starting point of the line.
  - **x2 and y2:** They define the end point of the line.
    - Also you can define as many points you want to.
  - CSS stroke: This attribute is compulsory in case of line as if you don't define stroke it will by default take the color of the line as white and the output will be something like this.

Output :



  - **CSS stroke-width:** This attribute is used to define the width of the line.

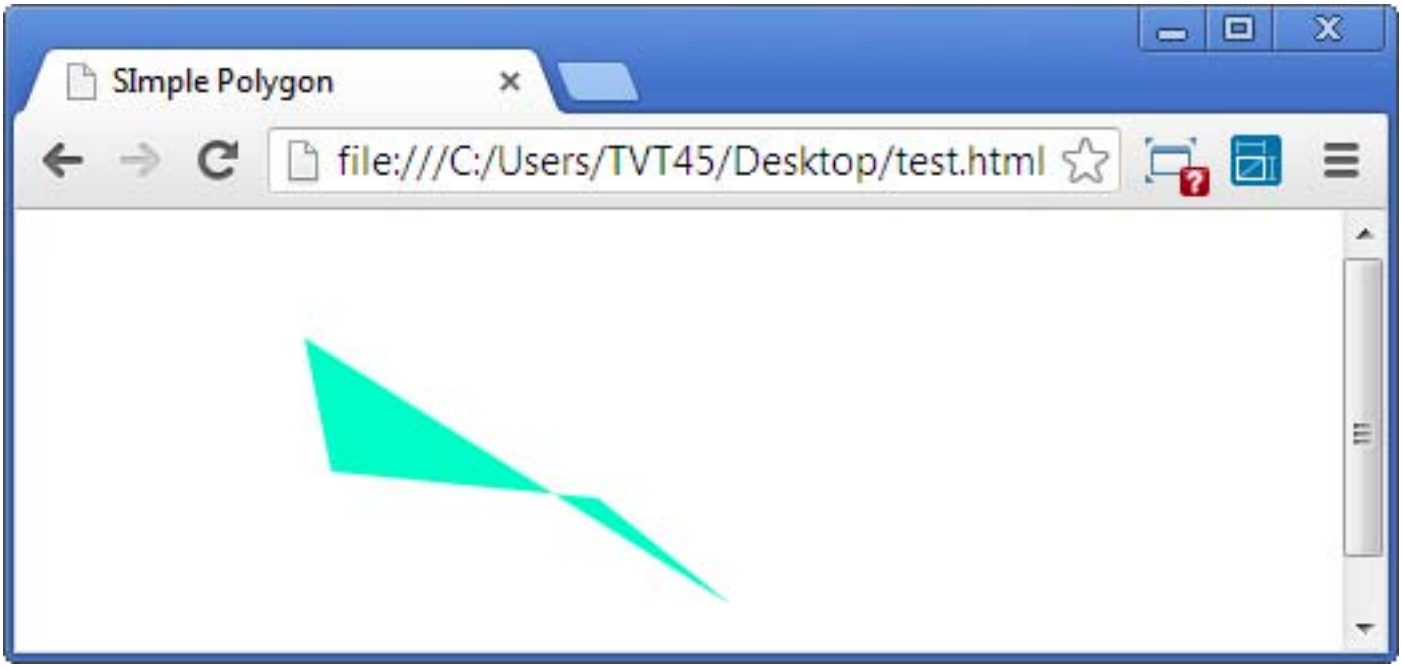
SVG POLYGON: <polygon> tag

- To create a polygon we use <polygon> tag.
- Example of a Simple polygon.

```
<!DOCTYPE html>
<html>
        <head>
                <title> SImple Polygon </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <polygon points="100,140,110,190,210,200,260,240"
```

```
style="fill:rgb(0,255,200)"/>
                        </svg>
            </body>
</html>
```

Output :



- The attributes used with <polygon> tag are:
  - **points:** This attribute is used to define the points of the line on the polygon. The shape of the polygon is completely
  - dependent on the points you define.

    CSS fill: This attribute is used to fill color in the polygon.

    CSS stroke: This attribute is used to define the color of the border of the polygon.

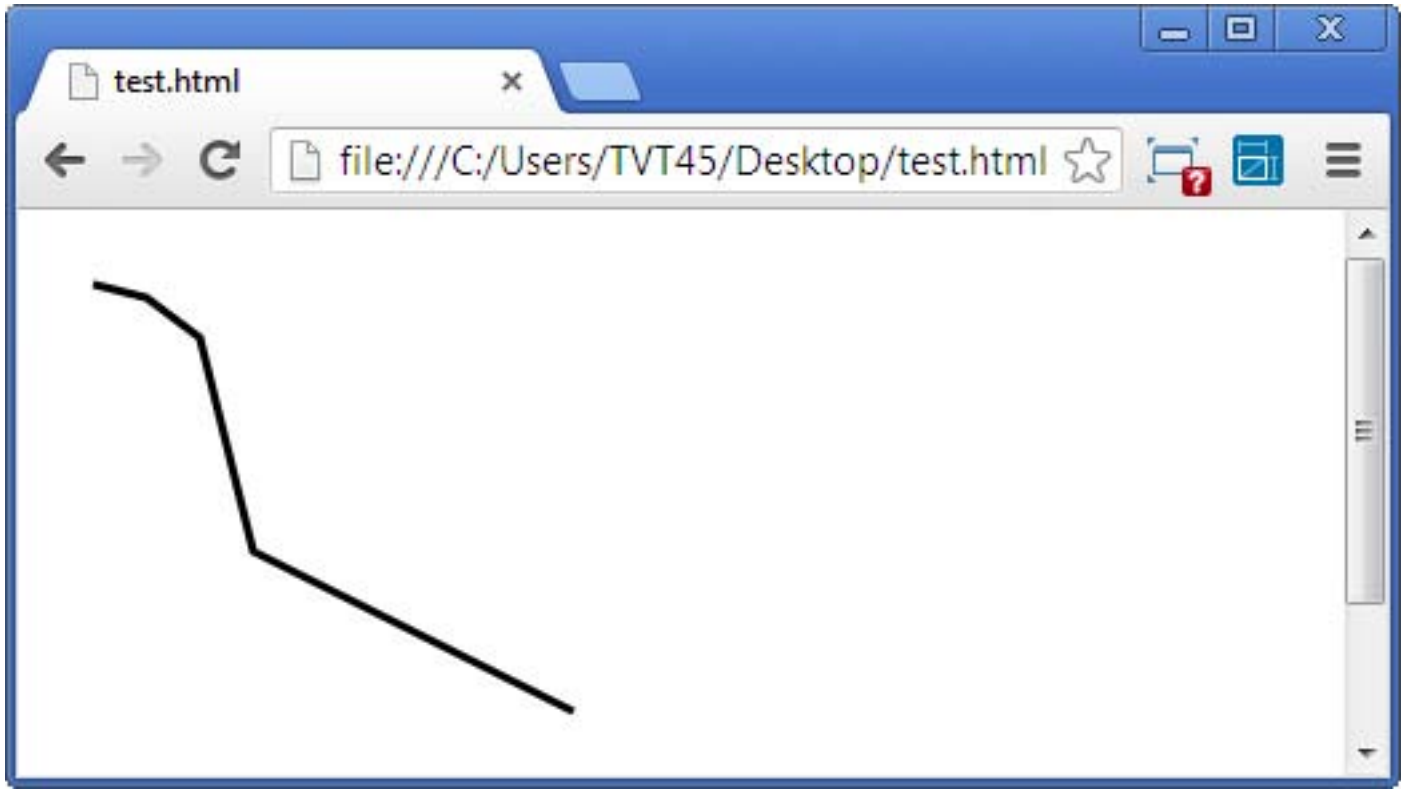
    CSS stroke-width: This attribute is used to define the width of the border of the polygon.

SVG POLYLINE: <polyline> tag

- <line> tag is used to create a single straight line. The <polyline> tag is used to create any shape that consists of only straight lines.
- Example of a Simple Polyline.

```
<!DOCTYPE html>
<html>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <polyline points="20,20 40,25 60,40 80,120 120,140 200,180" style="fill:none;stroke:black;stroke-width:3" />
</svg>
</body>
</html>
```

Output :



SVG PATH: <path> tag

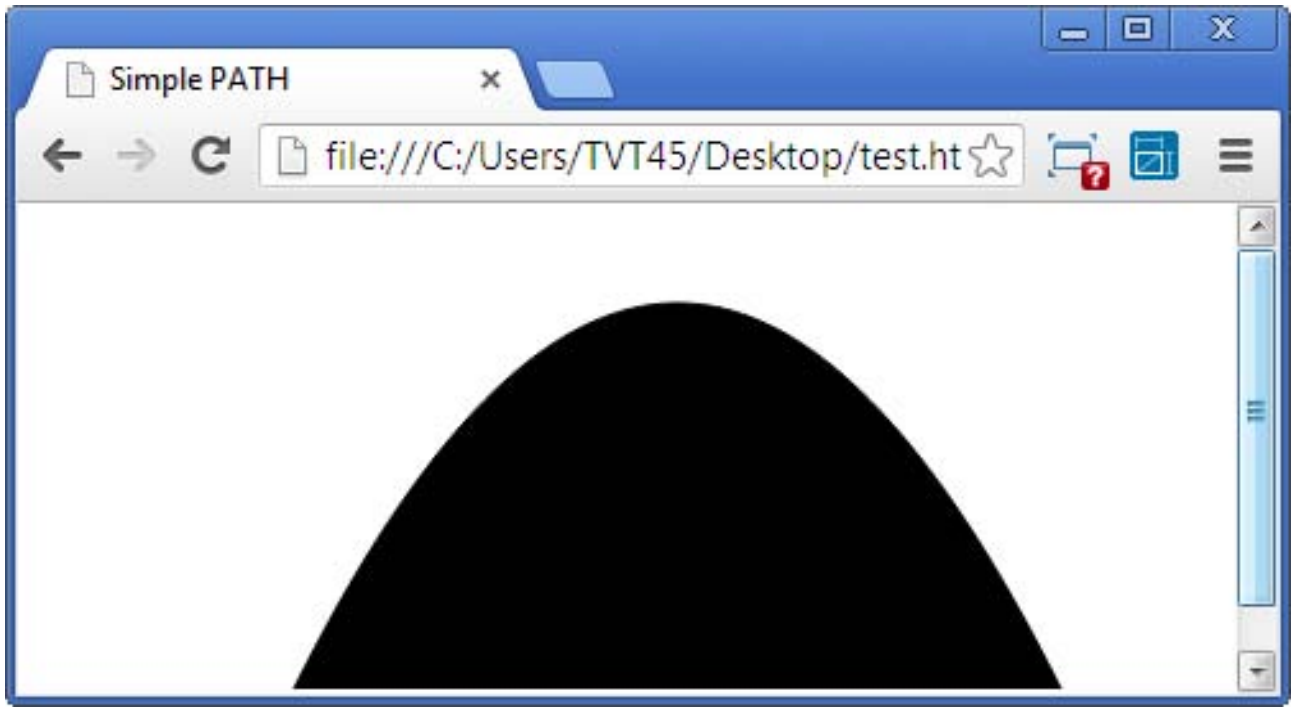
- <path> tag is used to create a specific path.
- <path> tag uses following commands to create a path.
  - M = moveto
  - L = lineto
  - H = horizontal lineto

- V = vertical lineto
- C = curveto
- S = smooth curveto
- Q = quadratic Bézier curve
- T = smooth quadratic Bézier curveto
- A = elliptical Arc
- Z = closepath

- Example of a Simple path.

```
<!DOCTYPE html>
<html>
        <head>
                <title> Simple PATH </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <path d="M 100 350 q 150 -300 300 0" style="stroke:black" />
                </svg>
        </body>
</html>
```

Output :



- Example of a Simple path.
  - **d:** It's the most important attribute of the <path> tag as it gives the direction in which the path is moving.
  - **CSS stroke:** This attribute is used to define the border. Also if this is not used nothing gets displayed.
  - **CSS fill:** This attribute is used to fill color in the shape created by the path.
  - **CSS stroke-width:** This attribute is used to define the width of the border.
- As coding becomes very complex in <path> tag, it is better to use SVG editors to create objects.
- Inkscape is one of the well known SVG editors.

SVG TEXT: <text> tag

- <text> tag is used to display text using SVG.
- Example of a Simple Text.

```
<!DOCTYPE html>
<html>
        <head>
                <title> Simple Text </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <text x="100" y="75" style="font-size:50"> Hey Guys!!! Its Party Time </text>
                </svg>
        </body>
</html>
```

Output :

SVG Stroke Attribute:

- SVG offers a wide range of stroke properties. They are:
  - stroke – To define the color of the border, stroke is used.

Example of stroke is:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Simple Stroke </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <line x1="100" y1="100" x2="100" y2="200" style="stroke:red" />
                </svg>
        </body>
</html>
```

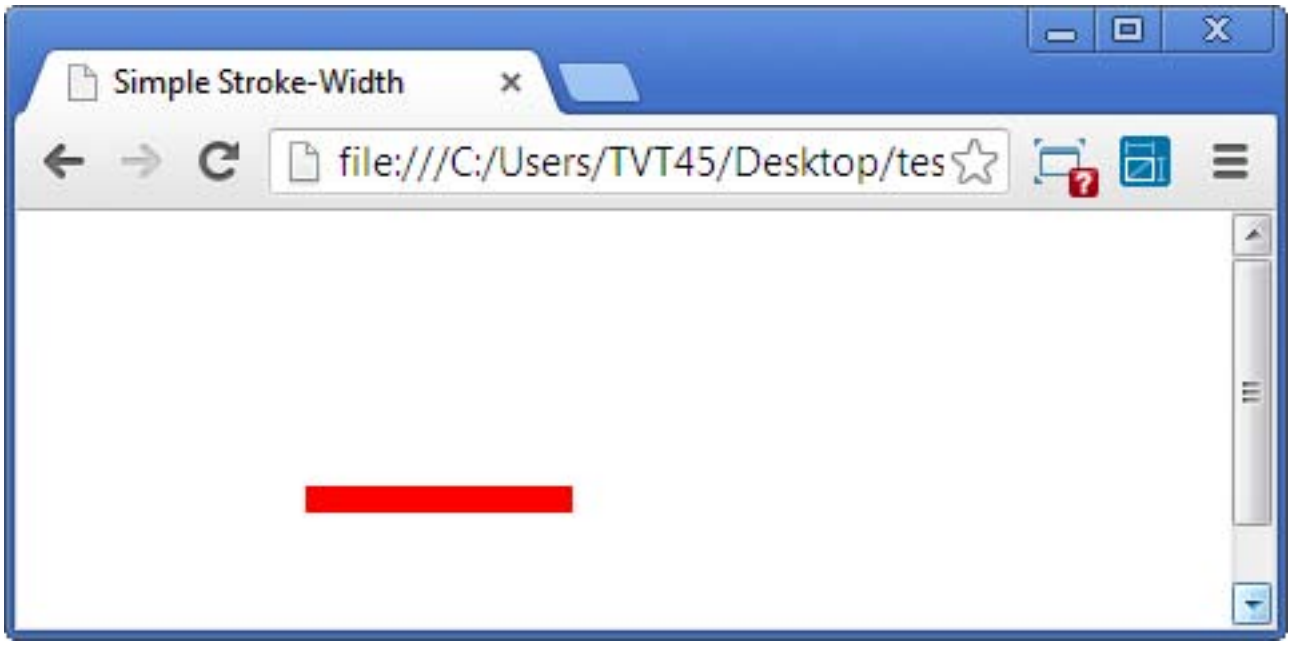
Output :

- stroke – To define the color of the border, stroke is used.

Example of stroke-width is:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Simple Stroke-Width </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <line  x1="100"  y1="100"  x2="200"  y2="100"  style="stroke:red;  stroke-
width:10" />
                </svg>
        </body>
</html>
```

Output :

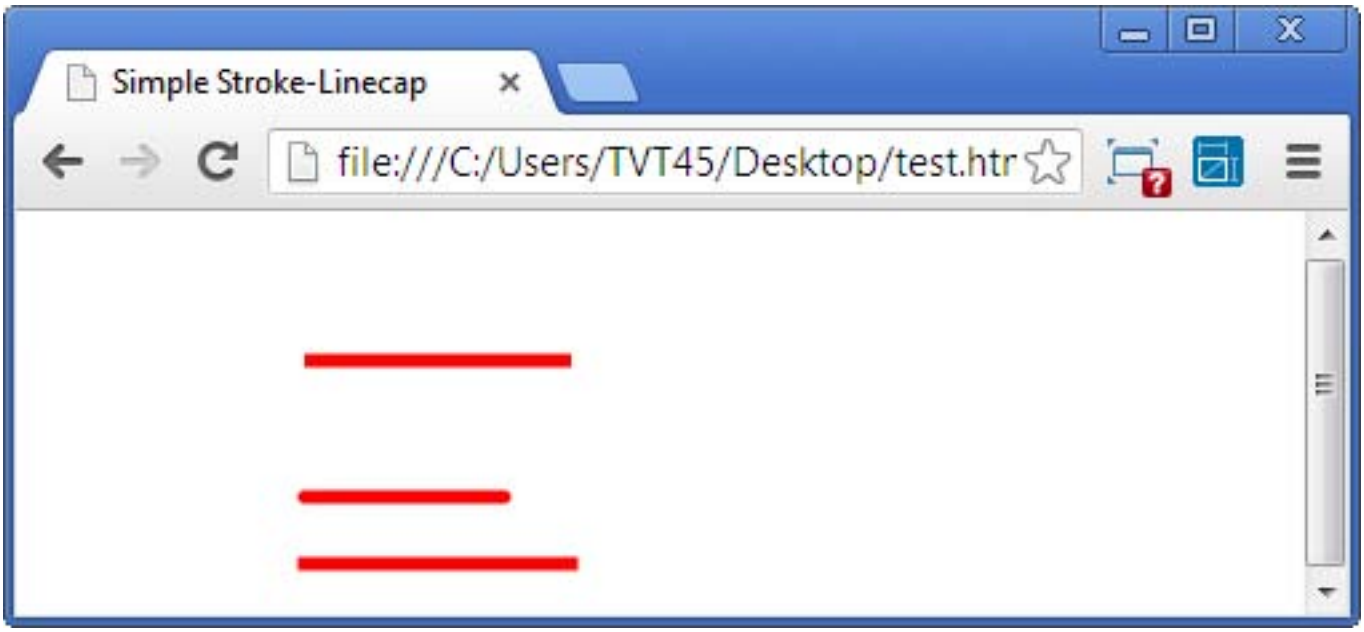


- stroke-linecap – To define different types of endings to an open path, stroke-linecap is used. The different types are 'butt', 'round' and 'square'.

Example of stroke-linecap:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Simple Stroke-Linecap </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <line  x1="100"  y1="100"  x2="200"  y2="100"  style="stroke:red;  stroke-
linecap:butt; stroke-width:5 " />
                        <line  x1="100"  y1="175"  x2="175"  y2="175"  style="stroke:red;  stroke-
linecap:round; stroke-width:5 " />
                        <line  x1="200"  y1="200"  x2="100"  y2="200"  style="stroke:red;  stroke-
linecap:square;stroke-width:5 " />
                </svg>
        </body>
</html>
```

Output :

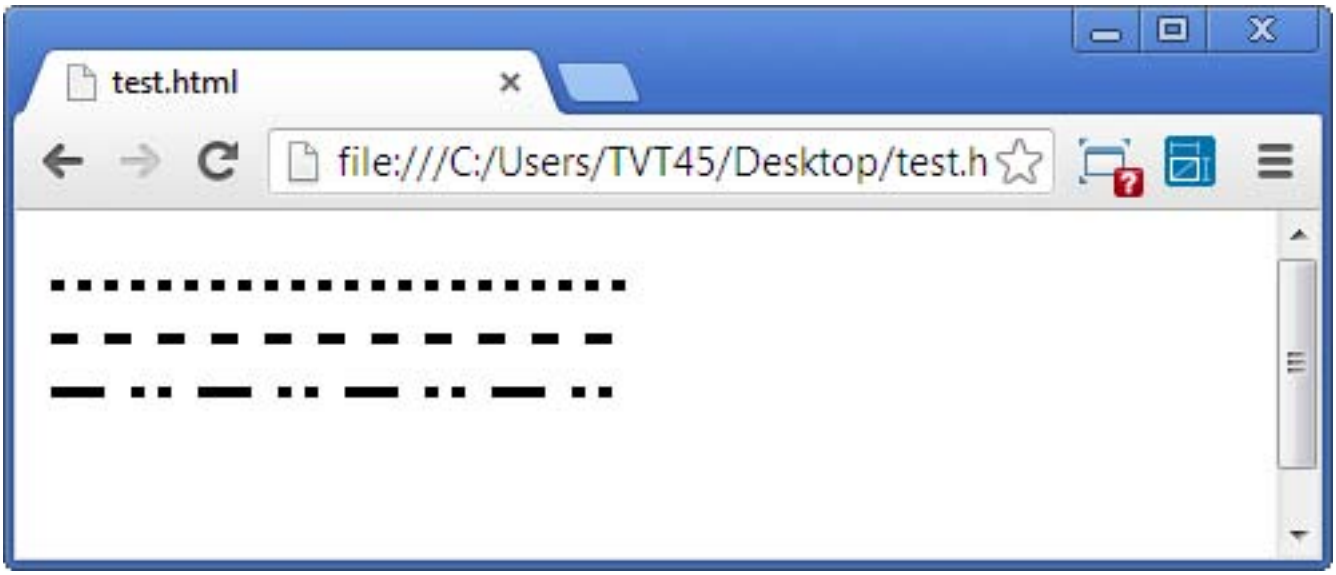


- stroke-dasharray – To define different types of dashed lines, stroke-dasharray is used.

Example of stroke-dasharray is:

```
<!DOCTYPE html>
<html>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                                        <g fill="none" stroke="black" stroke-width="4">
                                                <path stroke-dasharray="5,5" d="M5 20 l215 0" />
                                                <path stroke-dasharray="10,10" d="M5 40 l215 0" />
                                                <path stroke-dasharray="20,10,5,5,5,10" d="M5 60 l215 0" />
                                        </g>
</svg>
</body>
</html>
```

Output :



SVG Filters

- SVG filter effects are effects applied to Scalable Vector Graphics (SVG) files.
- A filter effect is a graphical operation that is applied to an element as it is drawn into the document.
- It is an image-based effect, in that it takes zero or more images as input, a number of parameters specific to the effect, and then produces an image as output.
- The output image is either rendered into the document instead of the original element, used as an input image to another filter effect, or provided as a CSS image value.

- A simple example of a filter effect is a "flood". It takes no image inputs but has a parameter defining a color. The effect produces an output image that is completely filled with the given color.
- SVG supports following filters:

1 feBlend
2 feColorMatrix
3 feComponentTransfer
4 feComposite
5 feConvolveMatrix
6 feDiffuseLighting
7 feDisplacementMap
8 feFlood
9 feGaussianBlur
10 feImage
11 feMerge
12 feMorphology
13 feOffset
14 feSpecularLighting
15 feTile
16 feTurbulence
17 feDistantLight
18 fePointLight
19 feSpotLight

SVG Filter Primitive <feBlend> and <feOffset>:

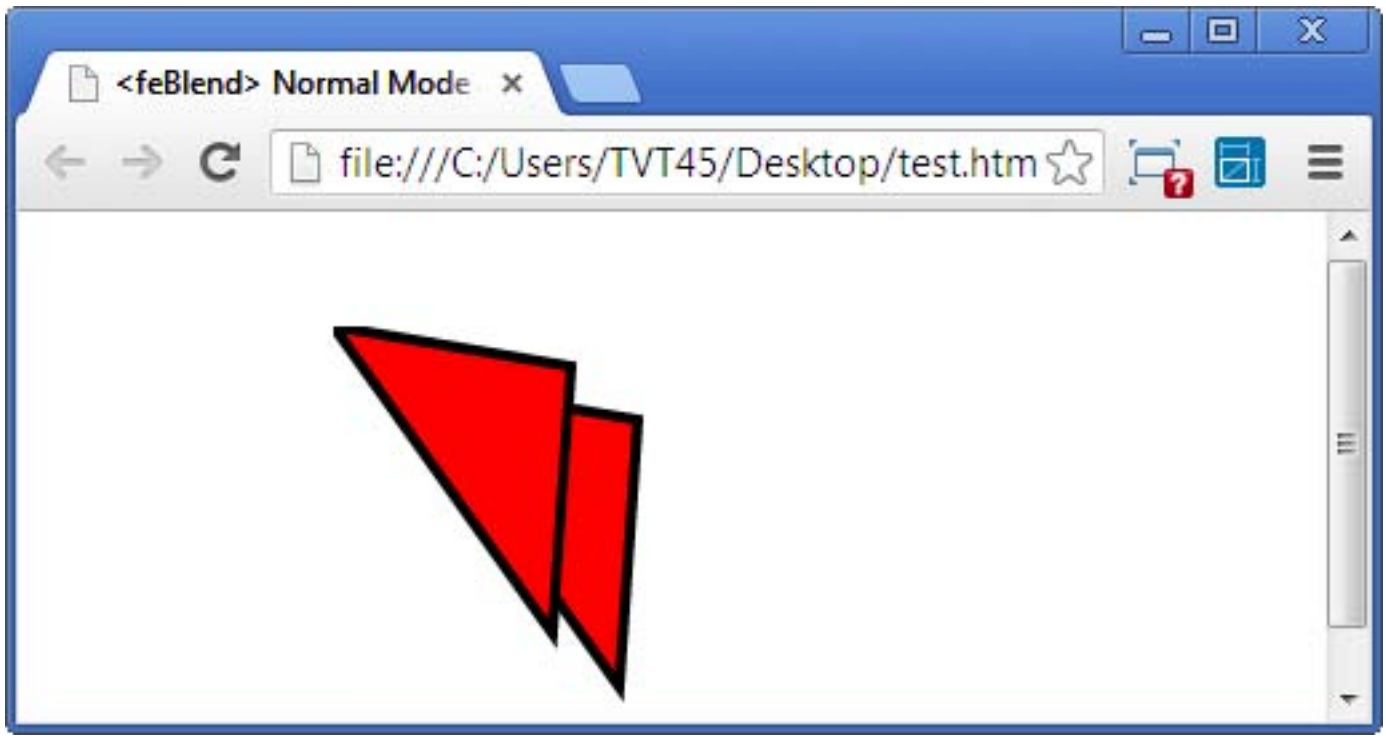
- As the name suggests it blends two or more images.SVG supports following filters:
- In a more technical way it can be defined as: "This filter composites two objects together using commonly used imaging software blending modes. It performs a pixel-wise combination of two input images."

- The modes of <feBlend> primitive are:
  - normal (default)
  - multiply
  - screen
  - darken
  - lighten
- The attributes of <feBlend> are:
  - **mode:** defines the mode of <feBlend>.
  - **in:** defines the first image input to the blending operation.
  - **in2:** define the second image input to the blending operation.
  - Then there are many presentation attributes.
  - **style:** used to define the inline CSS.

SVG Filter Primitive <feBlend> and <feOffset>:

```
<html>
        <head>
                <title> <feBlend> Normal Mode and <feSoffset> </title>
        </head>
<body>
 <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                <defs>
 <filter id="filter1" x="0" y="0" width="150%" height="150%">
 <feOffset result="offOut" in="SourceGraphic" dx="25" dy="20" />
 <feBlend in="SourceGraphic" in2="offOut" mode="normal" />
 </filter>
 </defs>
    <polygon points="110,135,192, 250, 199, 150,110" style="fill:red;stroke:black;stroke-width:4"
filter="url(#filter1)" />
                </svg>
</body>
</html>
```

Output :



SVG Filter Primitive <feGaussianBlur>:

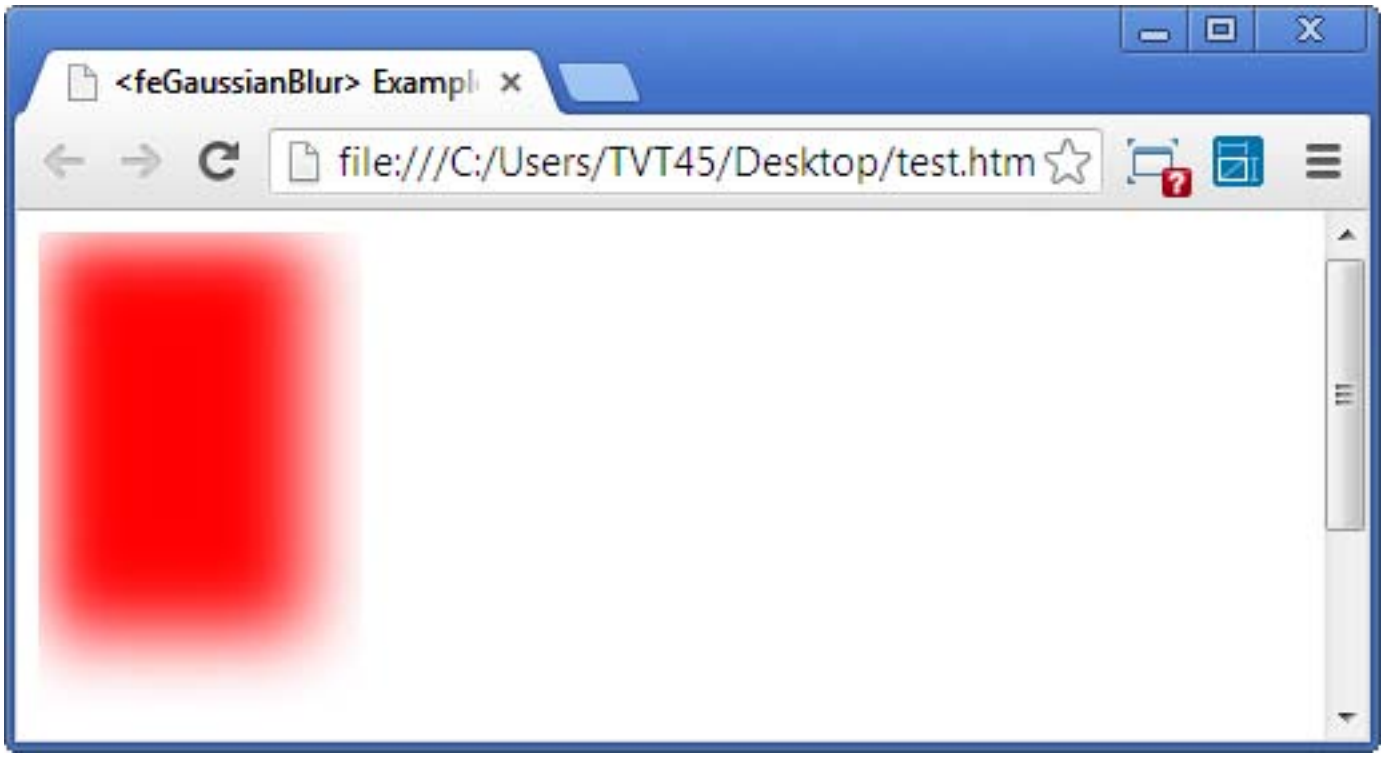
- The **<feGaussianBlur>** element is used to create blur effects.
- The attributes for **<feGaussianBlur>** are:
  - **in:** defines the image input to the blurring operation.
  - **stdDeviation:** The standard deviation for the blur operation. If two numbers are provided, the first number represents a standard deviation value along the x-axis of the coordinate system established by attribute 'primitive Units' on the <filter> element. The second value represents a standard deviation in Y. If one number is provided, then that value is used for both X and Y.

    A value of zero disables the effect of the given filter primitive.
  - There are many presentation attributes also.

Example for Filter Primitive <feGaussianBlur>:

```
<html>
        <head>
                <title> <feGaussianBlur> Example </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <defs>
                                <filter id="filter1" x="0" y="0">
                                        <feGaussianBlur                 in="SourceGraphic"
stdDeviation="14" />
                                </filter>
                        </defs>
                        <rect width="100" height="150" style="fill:rgb(255,0,0);" filter="url(#filter1)"
/>
                </svg>
        </body>
</html>
```

Output :



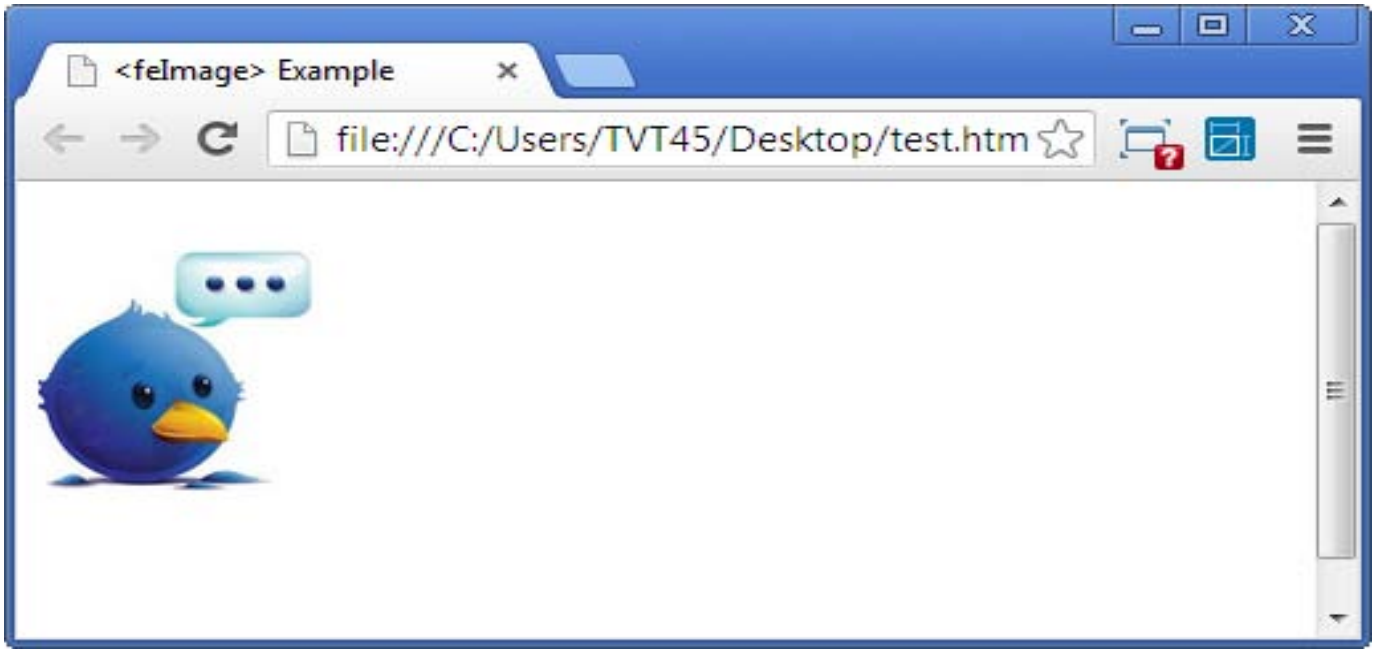
SVG Filter Primitive <feImage>

- This filter primitive can refer to an external image or can be a reference to another piece of SVG. It produces an image similar to the built-in image source Source Graphic except that the graphic comes from an external source.
- If the xlink:href references a stand-alone image resource such as a JPEG, PNG or SVG file, then the image resource is rendered according to the behavior of the <image> element;

Example For Filter Primitive <feImage> is:

```
<html>
        <head>
                <title> <feImage> Example </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <defs>
                                <filter id="Default">
                                        <feImage xlink:href="index.jpg" />
                                </filter>
                        </defs>
                        <rect width="100" height="150" filter="url(#Default)" />
                </svg>
        </body>
</html>
```

Output :



SVG Filter Primitive <feColorMatrix>:

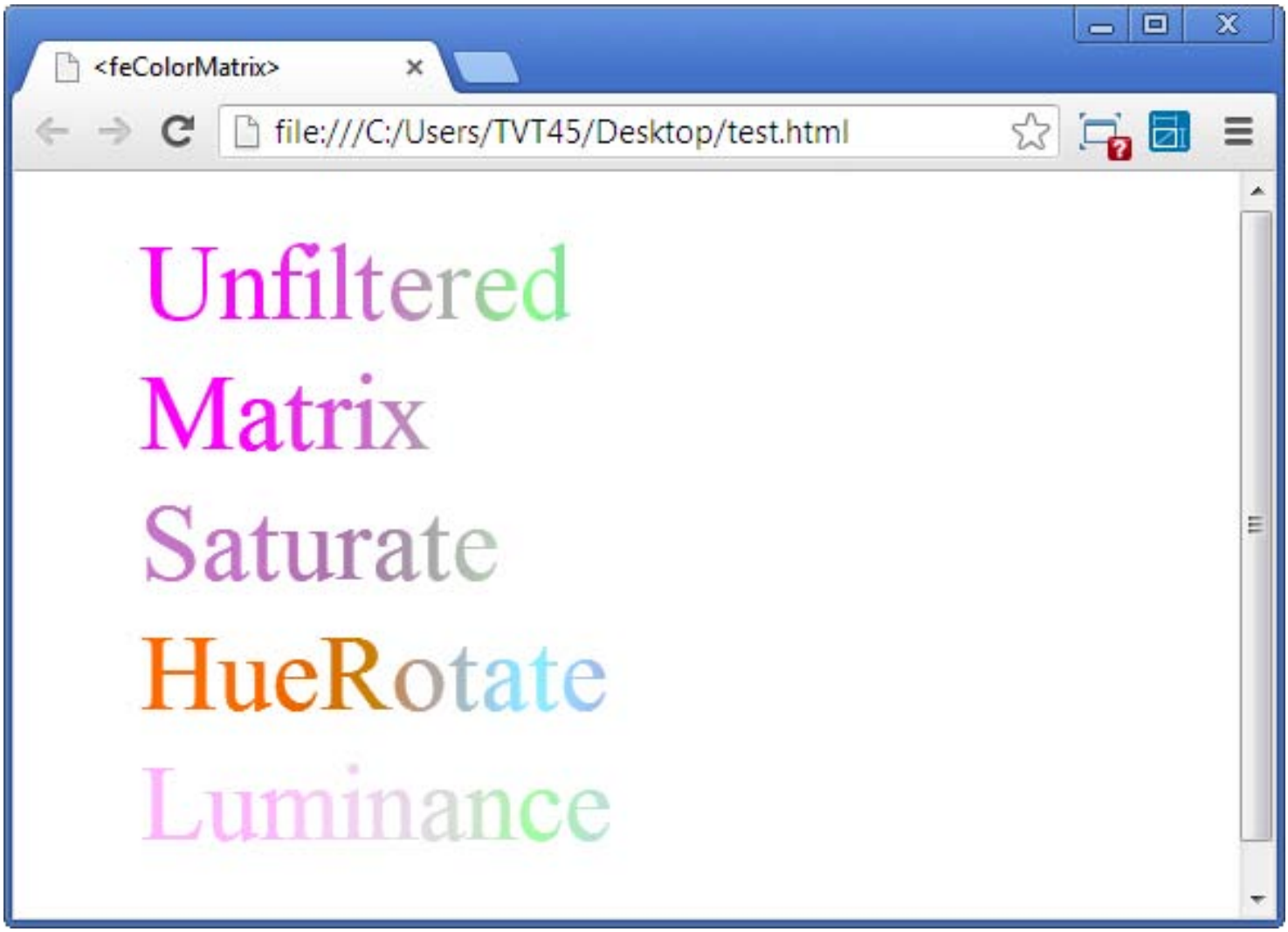
- The idea behind the feColorMatrix effect is to apply a matrix transform to the Red, Green, Blue and Alpha channels of each pixel in the source image.
- The feColorMatrix effect takes a type parameter which allows us to specify the full matrix or a shorthand value that will calculate the rest of the matrix values for us.
- **There are 4 types of color matrices:**
  - Matrix
  - Saturate
  - hueRotate
  - luminanceToAlpha
- **Type = matrix**
  - Setting the type to "matrix" gives us access to the entire matrix.
  - The matrix coefficients are given in row-major order in a string separated by white space (including end of line) or commas.
- **Type = saturate**
  - A matrix type of "saturate" adjusts the saturation of the colour using a real number value of 0 to 1 supplied as the "values" attribute of feColorMatrix.
- **Type = hueRotate**
  - The "hueRotate" type takes an angle (in degree) as the value of "values" and rotates the pixel hue by the given angle.
- **Type = luminanceToAlpha**
  - a type of "luminanceToAlpha" converts the red, green and blue channels into a luminance value then sets the output alpha channel based to the result.
  - The "values" attribute is not used with this type.

SVG Filter Primitive <feColorMatrix>:

Example of <feColorMatrix> with all the types:

```
<!DOCTYPE html>
<html>
        <head>
                <title> <feColorMatrix> </title>
        </head>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <defs>
                                <linearGradient id="MyGrad" gradientUnits="userSpaceOnUse"
x1="100" y1="0" x2="500" y2="0">
                                        <stop offset="0" style="stop-color:#ff00ff" />
                                        <stop offset=".33" style="stop-color:#88ff88" />
                                        <stop offset=".67" style="stop-color:#2020ff" />
                                        <stop offset="1" style="stop-color:#d00000" />
                                </linearGradient>
                                <filter id="Matrix">
                                        <feColorMatrix type="matrix" values="1 0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0 0 1 0" />
                                </filter>
                                <filter id="Saturate">
                                        <feColorMatrix type="saturate" values="0.4" />
                                </filter>
                                <filter id="HueRotate">
                                        <feColorMatrix type="hueRotate" values="90" />
                                </filter>
                                <filter id="Luminance">
                                        <feColorMatrix type="luminanceToAlpha" result="a"
/>
                                        <feComposite in="SourceGraphic" in2="a"
operator="in" />
                                </filter>
                        </defs>
                        <g style="font-size:50;fill:url(#MyGrad)">
                                <text x="50" y="60">Unfiltered</text>
                                <text x="50" y="120" filter="url(#Matrix)">Matrix</text>
                                <text x="50" y="180" filter="url(#Saturate)">Saturate</text>
                                <text x="50" y="240" filter="url(#HueRotate)">HueRotate</text>
                                <text x="50" y="300"
filter="url(#Luminance)">Luminance</text>
                        </g>
</svg>
</body>
</html>
```

Output :



SVG Filter Primitive <feMerge> , <feComposite>, <feFlood>:

- The SVG feMerge element is used to create image layers on top of each other.
- Each feMerge element can have any number of feMergeNode sub-elements.
- The SVG <feFlood> filter primitive creates a rectangle filled with the color and opacity values from properties 'flood-color' and 'flood-opacity'.
- The rectangle is as large as the filter primitive subregion established by the 'x', 'y', 'width' and 'height' attributes on the 'feFlood' element.
- The SVG <feComposite> filter performs the combination of the two input images pixel-wise in image space using one of the Porter-Duff [PORTERDUFF] compositing operations: over, in, atop, out, xor.

Example of SVG Filter <feMerge>, <feComposite> , <feFlood> and <feOffset>:

```
<!DOCTYPE html>
<html>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <defs>
                                <filter  id="test"  filterUnits="objectBoundingBox"  x="0"  y="0"  width="1.5"
height="4">
                                        <feOffset result="Off1" dx="15" dy="20" />
                                        <feFlood style="flood-color:#ff0000;flood-opacity:0.8" />
                                        <feComposite in2="Off1" operator="in" result="C1" />

<feOffset in="SourceGraphic" result="Off2" dx="30" dy="40" />
                                        <feFlood style="flood-color:#ff0000;flood-opacity:0.4" />
                                        <feComposite in2="Off2" operator="in" result="C2" />

<feMerge>
                                                <feMergeNode in="C2" />
                                                <feMergeNode in="C1" />
                                                <feMergeNode in="SourceGraphic" />
                                         </feMerge>
                                </filter>
                        </defs>
                        <text    x="30"    y="100"    style="font:36px    verdana    bold;fill:blue;filter:url(#test)">
Hello!</text>
</svg>
</body>
/html>
```

Output :

SVG Filter Primitive <feMorphology>:

- The SVG feMorphology element is used to "fattening" or "thinning" a source graphic.
- The operator attribute defines whether to erode or dilate the source graphic.
- The operator attribute can take one of the following values:
  - erode (thinning)
  - dilate (fattening)
- The radius attribute defines the radius for the operation.
- If two numbers are specified, the first number is the x-radius and the second number is the y-radius. If one number is specified, then that value is used for both x and y. Default is 0

Example for <feMorphology>

```
<!DOCTYPE html>
<html>
        <head>
                <title> <feMorphology> </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <defs><filter id="erode">
                                        <feMorphology operator="erode" in="SourceGraphic"  radius="1" />
                                </filter>
                <filter id="dilate"> orphology operator="dilate" in="SourceGraphic"  radius="4" />
                                </filter>
                        </defs>
                        <g font-size="60">
                        <text x="10" y="50" filter="url(#erode)"> ERODE </text>
                        <text x="10" y="100" filter="url(#dilate)"> DILATE </text>
                        </g>
                </svg>
        </body>
</html>
```

Output :

SVG Filter Primitive <feComponentTransfer>:

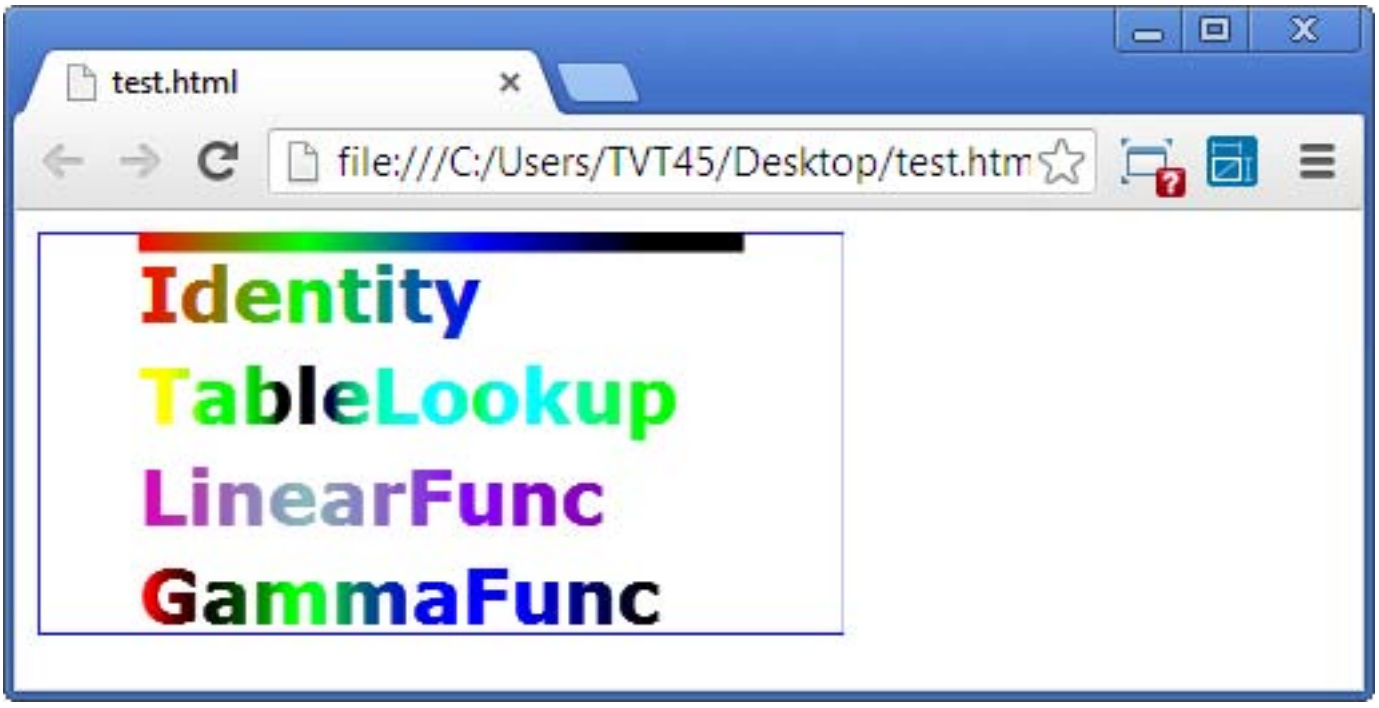
- This filter primitive performs component-wise remapping of data for every pixel.
- It allows operations like brightness adjustment, contrast adjustment, color balance or thresholding.
- The child elements of a 'feComponentTransfer' element specify the transfer functions for the four channels:
  - 'feFuncR' — transfer function for the red component of the input graphic
  - 'feFuncG' — transfer function for the green component of the input graphic
  - 'feFuncB' — transfer function for the blue component of the input graphic
  - 'feFuncA' — transfer function for the alpha component of the input graphic
- The following rules apply to the processing of the 'feComponentTransfer' element:
  - If more than one transfer function element of the same kind is specified, the last occurrence is to be used.
  - If any of the transfer function elements are unspecified, the 'feComponentTransfer' must be processed as if those transfer function elements were specified with their 'type' attributes set to 'identity'.
- The attributes of the sub elements are:
  - Type: Indicates the type of component transfer function.

Example for Filter Primitive <feComponentTransfer>:

```
<!DOCTYPE html>
<html>
<body>
<svg width="8cm" height="4cm" viewBox="0 0 800 400" xmlns = "http://www.w3.org/2000/svg"
version="1.1">
                        <defs>
 <linearGradient id="MyGradient" gradientUnits="userSpaceOnUse" x1="100" y1="0" x2="600" y2="0">
                                    <stop offset="0" stop-color="#ff0000" />
<stop offset=".33" stop-color="#00ff00" />
 <stop offset=".67" stop-color="#0000ff" />
 <stop offset="1" stop-color="#000000" />
 </linearGradient>
 <filter id="Identity" filterUnits="objectBoundingBox" x="0%" y="0%" width="100%" height="100%">
<feComponentTransfer>
<feFuncR type="identity"/>
 <feFuncG type="identity"/>
 <feFuncB type="identity"/>
<feFuncA type="identity"/>
</feComponentTransfer>
 </filter>
<filter id="Table" filterUnits="objectBoundingBox" x="0%" y="0%" width="100%" height="100%">
 <feComponentTransfer>
 <feFuncR type="table" tableValues="0 0 1 1"/>
 <feFuncG type="table" tableValues="1 1 0 0"/>
<feFuncB type="table" tableValues="0 1 1 0"/>
 </feComponentTransfer>
 </filter>
```

```
<filter id="Linear" filterUnits="objectBoundingBox" x="0%" y="0%" width="100%" height="100%">
<feComponentTransfer>
<feFuncR type="linear" slope=".5" intercept=".25"/>
<feFuncG type="linear" slope=".5" intercept="0"/>
<feFuncB type="linear" slope=".5" intercept=".5"/>
</feComponentTransfer>
</filter>
<filter id="Gamma" filterUnits="objectBoundingBox" x="0%" y="0%" width="100%" height="100%">
<feComponentTransfer>
        <feFuncR type="gamma" amplitude="2" exponent="5" offset="0"/>
<feFuncG type="gamma" amplitude="2" exponent="3" offset="0"/>
<feFuncB type="gamma" amplitude="2" exponent="1" offset="0"/>
</feComponentTransfer>
</filter>
                        </defs>
                        <rect fill="none" stroke="blue" x="1" y="1" width="798" height="398"/>
<g font-family="Verdana" font-size="75" font-weight="bold" fill="url(#MyGradient)" >
<rect x="100" y="0" width="600" height="20" />
<text x="100" y="90">Identity</text>
<text x="100" y="190" filter="url(#Table)" >TableLookup</text>
<text x="100" y="290" filter="url(#Linear)" >LinearFunc</text>
<text x="100" y="390" filter="url(#Gamma)" >GammaFunc</text>
                        </g>
</svg>
</body>
</html>
```

Example for Filter Primitive <feComponentTransfer>:



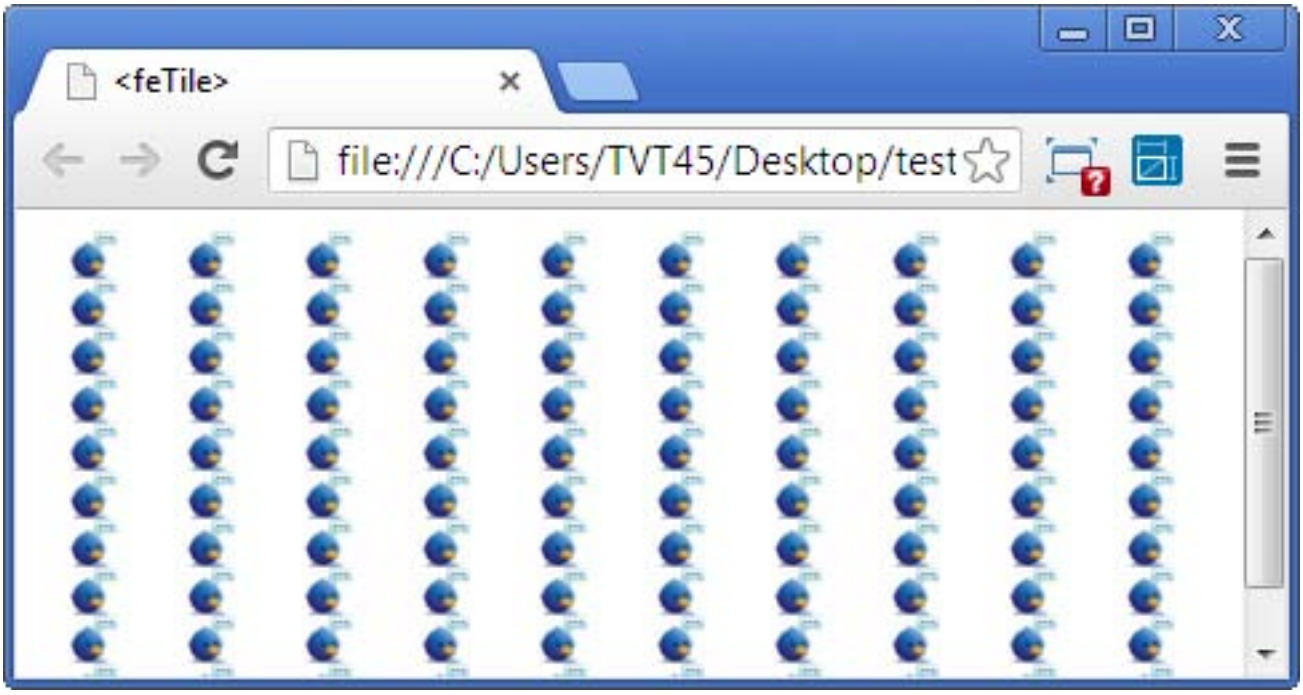
SVG Filter Primitive <feTile>:

- The 'feTile' filter effect can be used with an input to create a tiled pattern that fills the filter area.

Example of SVG Filter <feTile>:

```
<!DOCTYPE html>
<html>
	<head>
		<title> <feTile> </title>
	</head>
	<body>
		<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
			<filter id = "i1">
				<feImage x = "0" y = "0" width = "10%" height = "10%" result = "raster1" xlink:href
= "index.jpg"/>
				<feTile/>
			</filter>
			<g>
				<rect x = "0" y = "0" width = "100%" height = "100%" filter = "url(#i1)"/>
			</g>
		</svg>
	</body>
</html>
```

Output :



SVG Filter Primitive <fePointLight> , <feSpotLight> and <feDistantLight>:

- These filter primitives define light source as a point or a spot or light source from a distance.
- The attributes of the <fePointLight> are x , y and z that defines the position of the light source in respect to x, y and z direction.
- The attributes of the <feSpotLight> are:
    - 'x' , 'y' , 'z' defines the position of the light source with respect to x , y and z direction resp.
    - 'pointsAtX' defines X location in the coordinate system established by attribute 'primitiveUnits' on the 'filter' element of the point at which the light source is pointing.
    - 'pointsAtY' defines Y location in the coordinate system established by attribute 'primitiveUnits' on the 'filter' element of the point at which the light source is pointing.
    - 'pointsAtZ'
    - 'specularExponent' defines Exponent value controlling the focus for the light source.
    - 'limitingConeAngle' defines A limiting cone which restricts the region where the light is projected. No light is projected outside the cone. 'LimitingConeAngle' represents the angle in degrees between the spot light axis and the spot light cone.
- The attributes of the <feDistantLight> are
    - Azimuth: Direction angle for the light source on the XY plane (clockwise), in degrees from the x axis.
    - Elevation: Direction angle for the light source from the XY plane towards the z axis, in degrees.

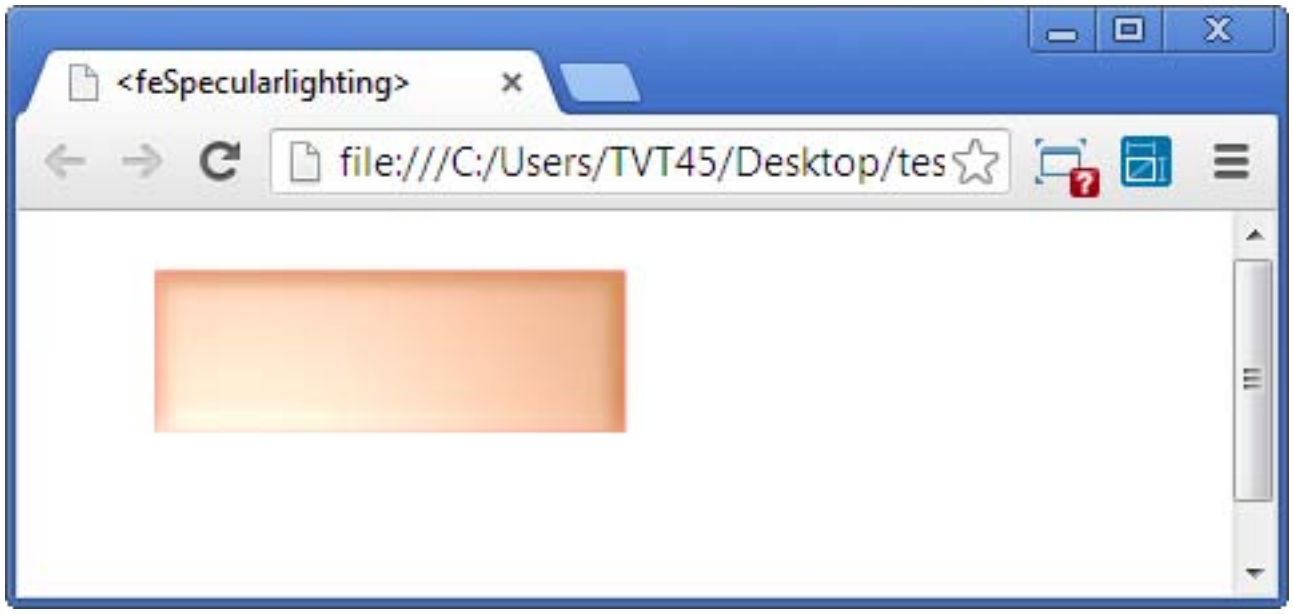
SVG Filter Primitive <feSpecularLighting>:

- This filter primitive lights a source graphic using the alpha channel as a bump map. The resulting image is an RGBA image based on the light color.

Example for Filter <feSpecularLighting> and <fePointLight>:

```
<!DOCTYPE html>
<html>
	<head>
		<title> <feSpecularlighting> </title>
	</head>
	<body>
		<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
			<defs>
				<filter id = "i1">
					<feGaussianBlur in = "SourceAlpha" stdDeviation = "4" result = "blur1"/>
					<feSpecularLighting result = "specOut" in = "blur1" specularExponent =
"20" lighting-color = "#bbbbbb">
						<fePointLight x = "50" y = "100" z = "200"/>
					</feSpecularLighting>
					<feComposite in = "SourceGraphic" in2 = "specOut" operator = "arithme-
tic" k1 = "0" k2 = "1" k3 = "1" k4 = "0"/>
				</filter>
			</defs>
			<g stroke = "tomato" fill = "peru" filter = "url(#i1)">
				<rect x = "10%" y = "10%" width = "40%" height = "40%"/>
			</g>
		</svg>
	</body>
</html>
```

Output:

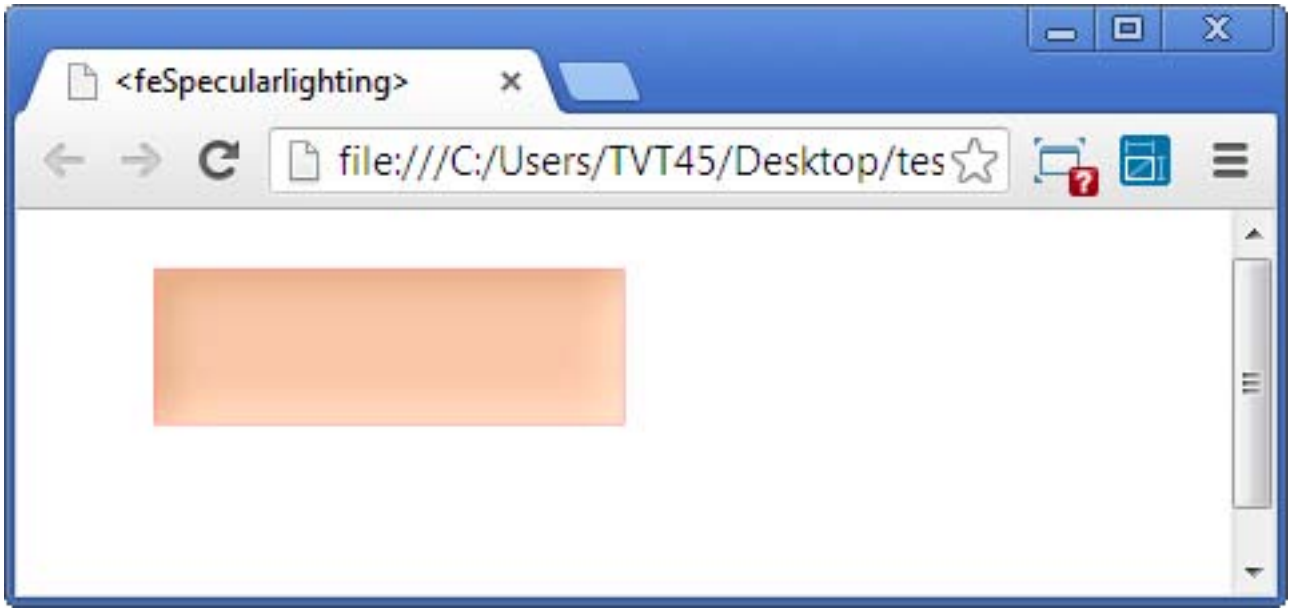


Example For <feSpecularLighting> and <feDistantLight>:

```
<!DOCTYPE html>
<html>
        <head>
                <title> <feSpecularlighting> </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <defs>
                                <filter id = "i1">
                                        <feGaussianBlur in = "SourceAlpha" stdDeviation = "8" result = "blur1"/>
                                        <feSpecularLighting result = "specOut" in = "blur1" specularConstant =
"1.2" specularExponent = "12" lighting-color = "#bbbbbb">
                                                <feDistantLight azimuth = "45" elevation = "45"/>
                                        </feSpecularLighting>
                                        <feComposite in = "SourceGraphic" in2 = "specOut" operator = "arithme-
tic" k1 = "0" k2 = "1" k3 = "1" k4 = "0"/>
```

```
</filter>
</defs>
                    <g stroke = "tomato" fill = "peru" filter = "url(#i1)">
                        <rect x = "10%" y = "10%" width = "40%" height = "40%"/>
                    </g>
                s</svg>
        </body>
</html>
```

Output:



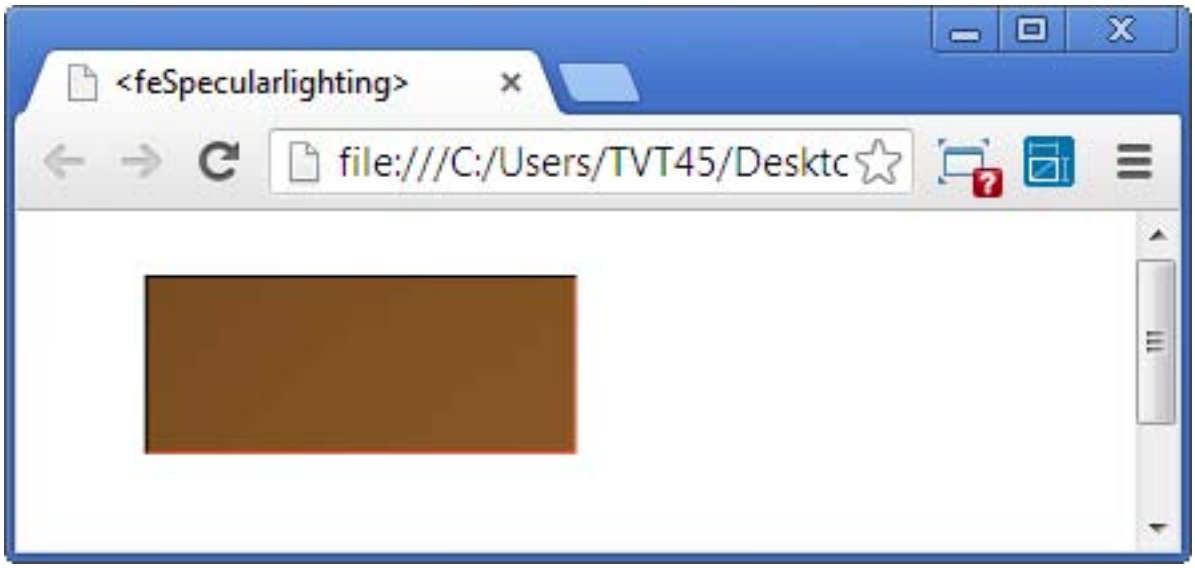
SVG Filter Primitive <feDiffuseLighting>:

- This filter primitive lights an image using the alpha channel as a bump map.

Example for <feDiffuseLighting> and <fePointLight>:

```
<body>
        <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
            <defs>
                <filter id = "i1">
                    <feDiffuseLighting result = "diffOut" in = "SourceGraphic" diffuseConstant
= "1.2" lighting-color = "white">
                        <fePointLight x = "400" y = "400" z = "150" pointsAtX = "0" point-
sAtY = "0" pointsAtZ = "0"/>
                    </feDiffuseLighting>
                    <feComposite in = "SourceGraphic" in2 = "diffOut" operator = "arithmetic"
k1 = "1" k2 = "0" k3 = "0" k4 = "0"/>
                </filter>
</defs>
            <g stroke = "tomato" fill = "peru" filter = "url(#i1)">
                <rect x = "10%" y = "10%" width = "40%" height = "40%"/>
            </g>
        </svg>
    </body>
</html>
```

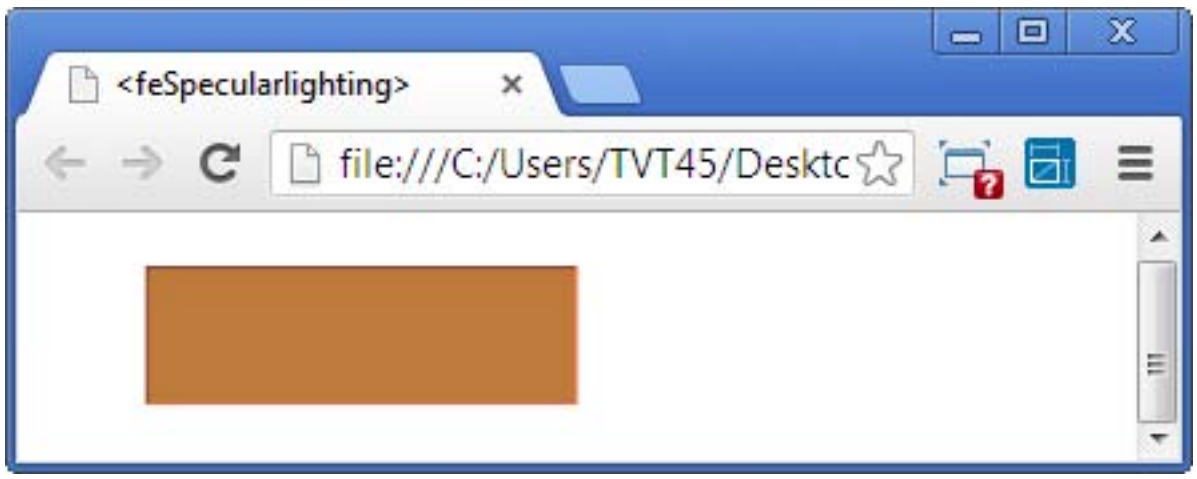
Output:



Example for <feDiffuseLighting> with <feDistantLight>

```
<!DOCTYPE html>
<html>
	<head>
		<title> <feSpecularlighting> </title>
	</head>
	<body>
		<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
			<defs>
				<filter id = "i1">
					<feDiffuseLighting result = "diffOut" in = "SourceGraphic" diffuseConstant
= "1.2" lighting-color = "white">
						<feDistantLight azimuth = "45" elevation = "45"/>
					</feDiffuseLighting>
					<feComposite in = "SourceGraphic" in2 = "diffOut" operator = "arithmetic"
k1 = "1" k2 = "0" k3 = "0" k4 = "0"/>
				</filter>
</defs>
			<g stroke = "tomato" fill = "peru" filter = "url(#i1)">
				<rect x = "10%" y = "10%" width = "40%" height = "40%"/>
			</g>
		</svg>
	</body>
</html>
```

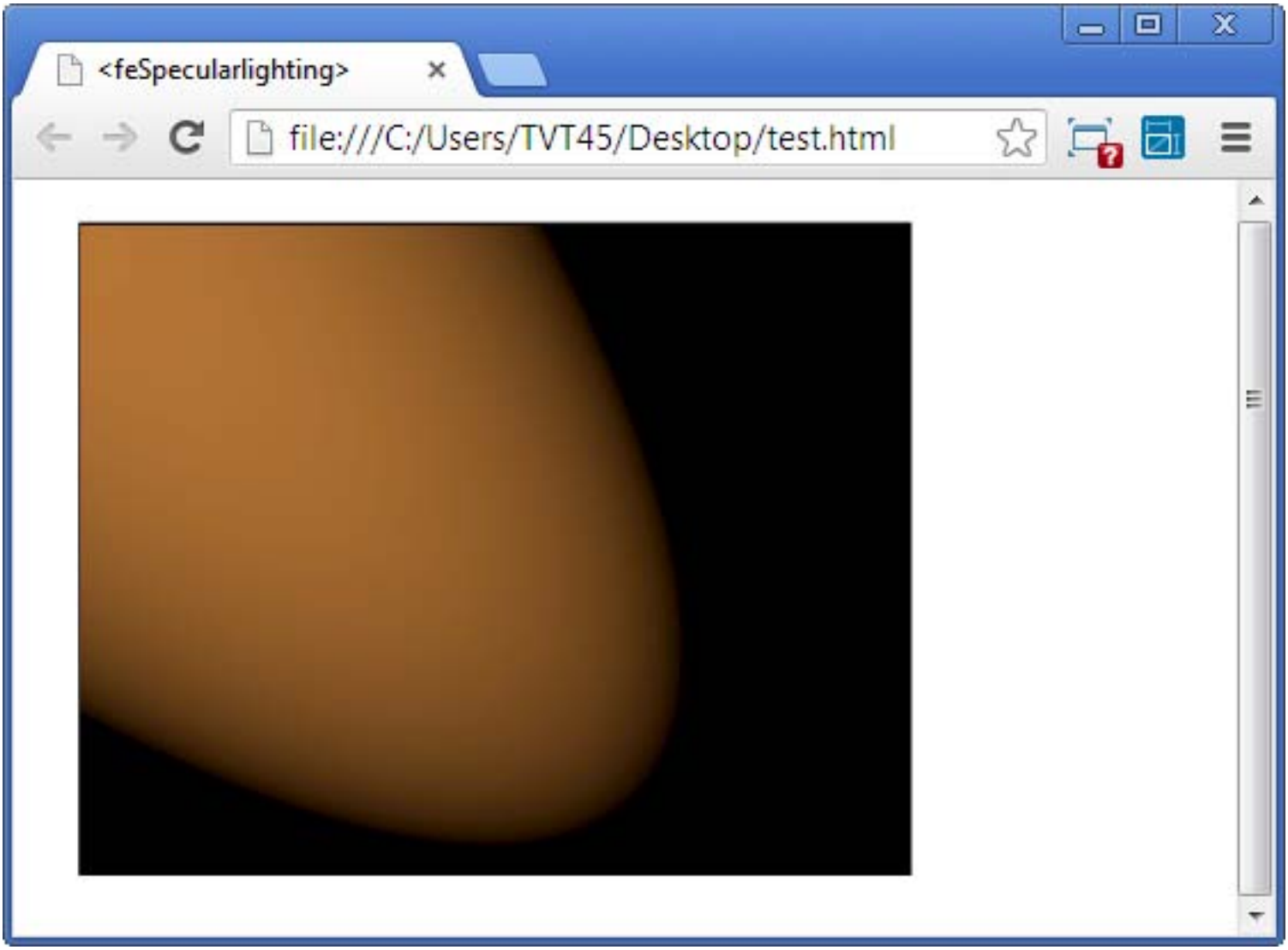
Output:



Example of <feDiffuseLighting> and <feSpotLight>

```
<!DOCTYPE html>
<html>
        <head>
                <title> <feSpecularlighting> </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <defs>
                                <filter id = "i1">
    <feGaussianBlur in = "SourceAlpha" stdDeviation = "8" result = "blur1"/>
    <feDiffuseLighting result = "diffOut" in = "SourceGraphic" diffuseConstant = "20" lighting-color = "white">
      <feSpotLight x = "800" y = "800" z = "400" pointsAtX = "0" pointsAtY = "0" pointsAtZ = "0"
limitingConeAngle = "9"/>
    </feDiffuseLighting>
    <feComposite in = "SourceGraphic" in2 = "diffOut" operator = "arithmetic" k1 = "1" k2 = "0" k3 = "0" k4 =
"0"/>
  </filter>
</defs>
  <g stroke = "tomato" fill = "peru" filter = "url(#i1)">
    <rect x = "10%" y = "10%" width = "40%" height = "40%"/>
  </g>
                </svg>
        </body>
</html>
```

Output:



SVG Gradients:

- A gradient is a gradual transition from one color to another. A filter effect is a graphical operation that is
- Gradients come in two basic forms: linear and radial. The type of gradient applied is determined by the element you use.

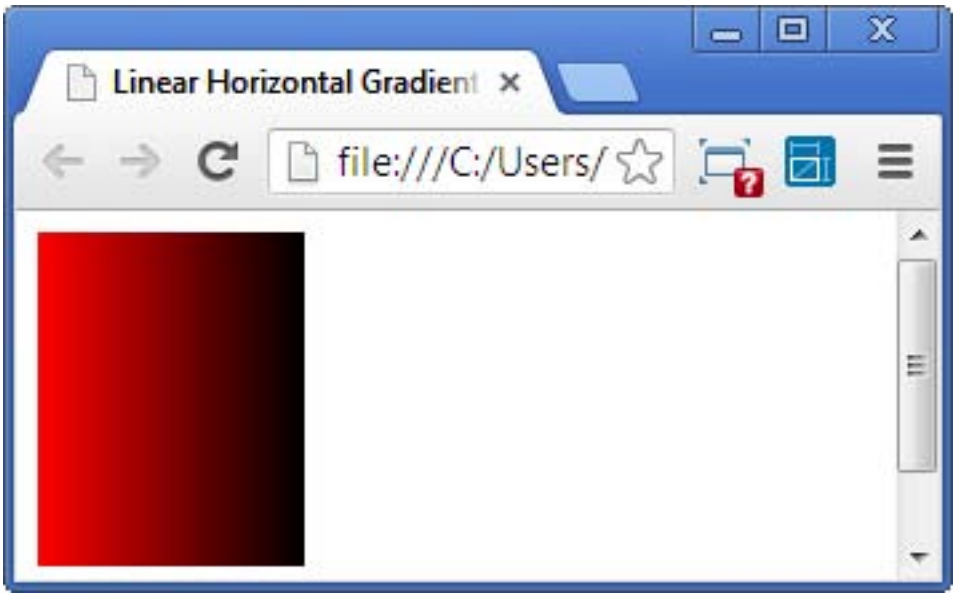
SVG Linear Gradient <linearGradient>:

- The <linearGradient> element is used to define a linear gradient.
- The <linearGradient> element must be nested within a <defs> tag. The <defs> tag is short for definitions and contains definition of special elements (such as gradients).
- Linear gradients can be defined as horizontal, vertical or angular gradients:
  - Horizontal gradients are created when y1 and y2 are equal and x1 and x2 differ.
  - Vertical gradients are created when x1 and x2 are equal and y1 and y2 differ.
  - Angular gradients are created when x1 and x2 differ and y1 and y2 differ.

Example For Linear Horizontal Gradients:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Linear Horizontal Gradient </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <defs><linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
                                        <stop offset="0%"  style="stop-color:rgb(255,0,0);stop-opacity:1" />
                                        <stop offset="100%"  style="stop-color:rgb(0,0,0);stop-opacity:1" />
                                </linearGradient>
                        </defs> <rect width="100" height="125" fill="url(#grad1)" />
                </svg>
        </body>
</html>
```

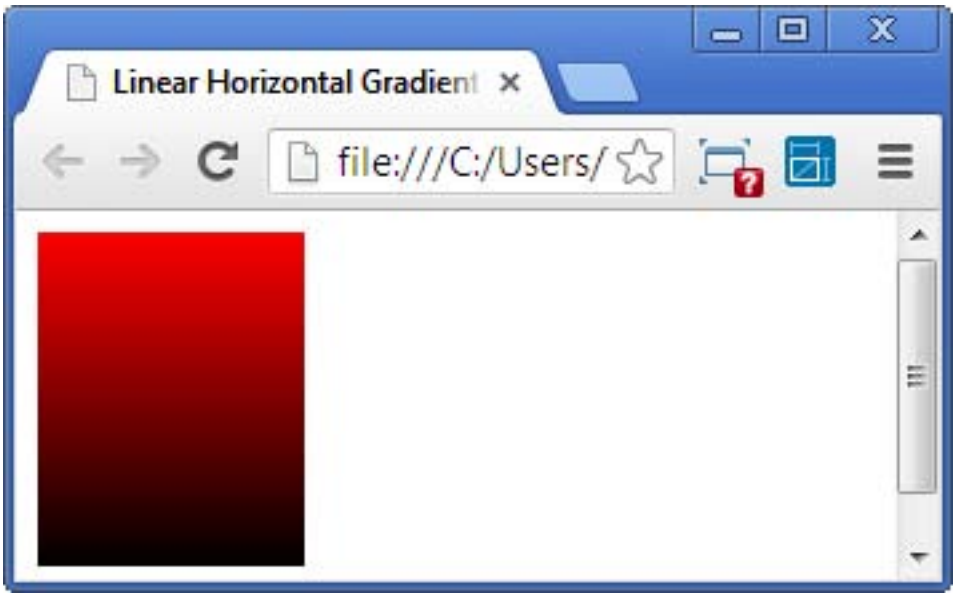
Output:



Example For Linear Vertical Gradients:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Linear Horizontal Gradient </title>
        </head>
        <body>
                <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                        <defs>
                                <linearGradient id="grad1" x1="100%" y1="0%" x2="100%" y2="100%">
                                        <stop offset="0%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
                                        <stop offset="100%" style="stop-color:rgb(0,0,0);stop-opacity:1" />
                                </linearGradient>
                        </defs>
                        <rect width="100" height="125" fill="url(#grad1)" />
                </svg>
        </body>
</html>
```

Output:



SVG Radial Gradient <radialGradient>:

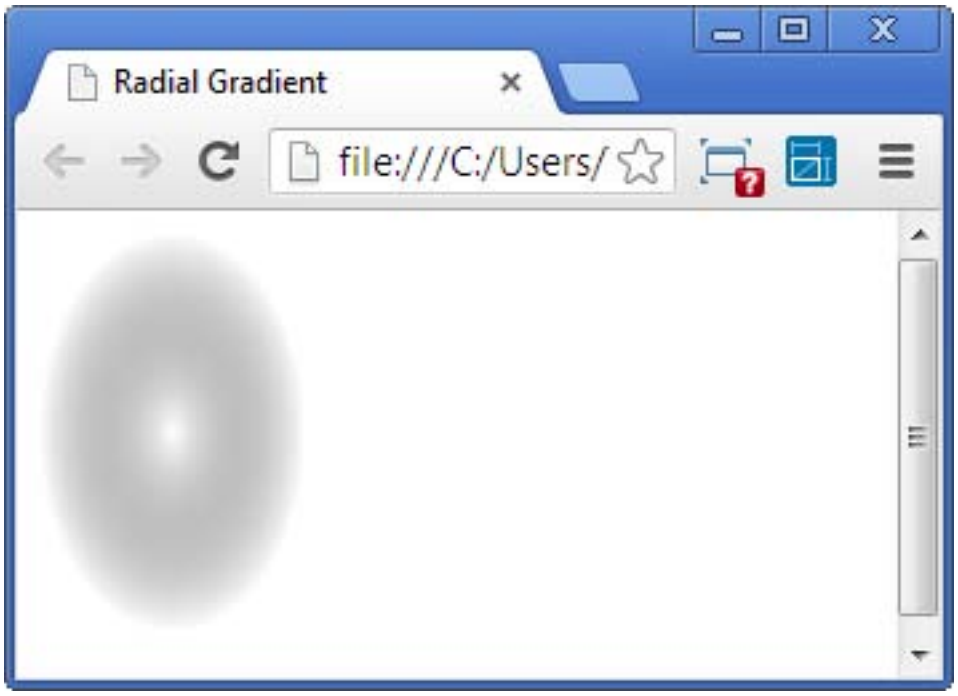
- The <radialGradient> element is used to define a radial gradient.
- The <radialGradient> element must be nested within a <defs> tag. The <defs> tag is short for definitions and contains definition of special elements (such as gradients).

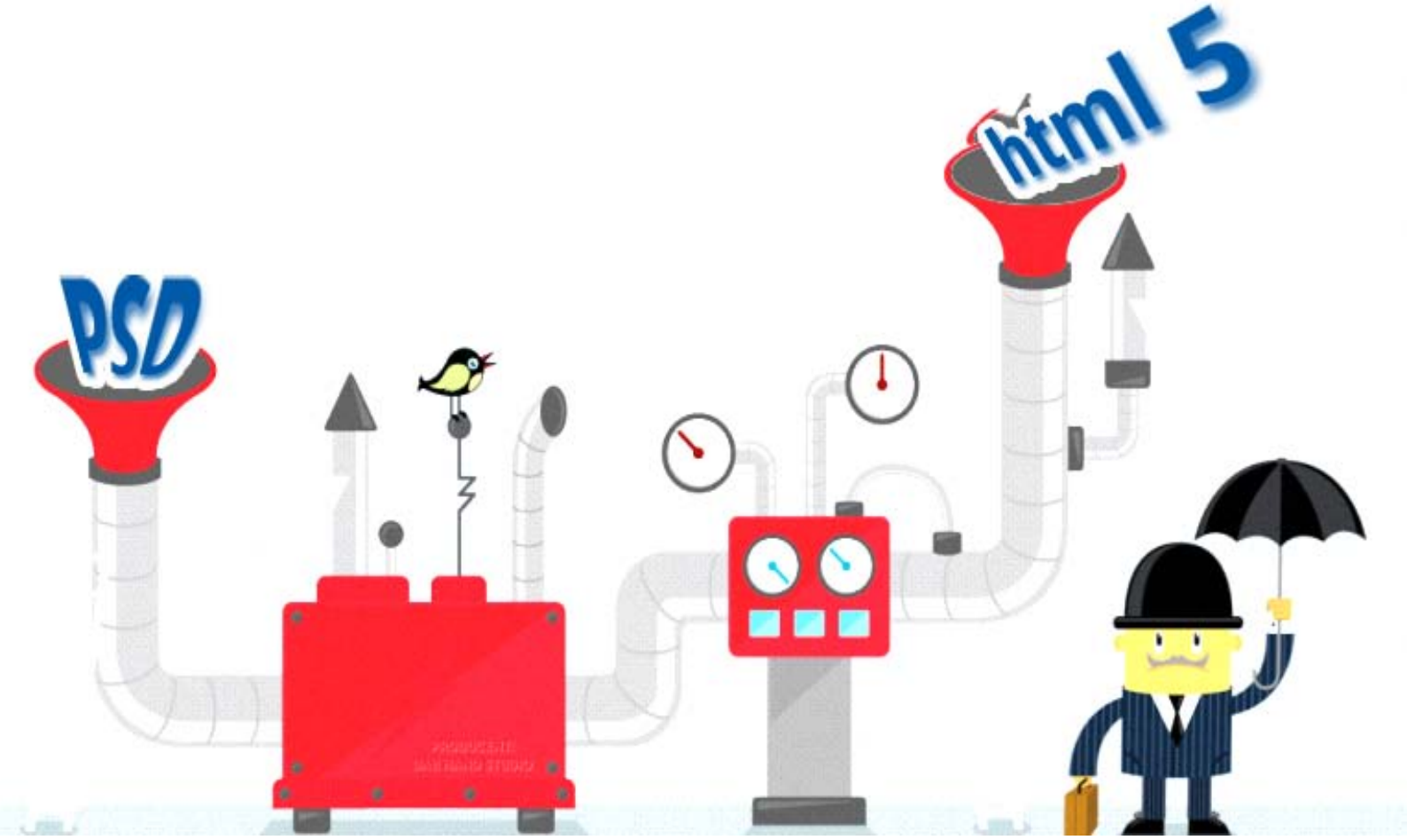
Example For <radialGradient>

```
<!DOCTYPE html>
<html>
        <head>
                <title> Radial Gradient </title>
        </head>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
                <defs>
  <radialGradient id="grad1" cx="50%" cy="50%" r="50%" fx="50%" fy="50%">
                                <stop offset="0%" style="stop-color:rgb(0,0,0);stop-opacity:0" />
                                <stop offset="100%" style="stop-color:rgb(255,255,255);stop-opacity:1" />
```

```
</radialGradient>
</defs>
        <rect width="100" height="150" fill="url(#grad1)" />
</svg>
</body>
</html>
```

Output:




PSD
html 5

MULTIMEDIA AND HTML5

- With HTML5 we can play video and audio very easily.
- There are 2 elements in HTML5 That come under multimedia
  - <audio>
  - <video>

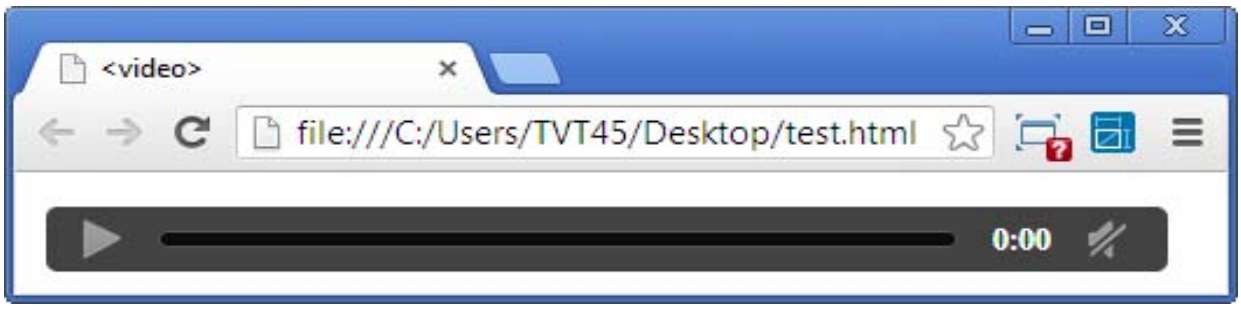
<video> tag

- Today, most videos are shown through a plug-in (like flash). However, different browsers may have different plug-ins.
- HTML5 defines a new element which specifies a standard way to embed a video/movie on a web page: the <video> element.
-
  - <source> : Defines the source of the video.
  - <track> : Defines text tracks in media players

Example of a <video> tag:

```
<!DOCTYPE html>
<html>
        <head>
                <title> <video> </title>
        </head>
        <body>
                <video height="300" width="450" controls>
                        <source src="video1.mp4" type="video/">
                </video>
        </body>
</html>
```

Output:



- The attributes of the <video> tag are:
  - Height: Defines the height of the window in which the video is displayed.
  - Width: Defines the width of the window in which the video is displayed.
  - Control: Displays the controls like: PAUSE, PLAY and VOLUME.

- The attributes of the <source> tag are:
  - Src: Defines the source of the video file.
  - Type: Defines the MIME type of the video like video/mp4 or video/ ogg or video/webM.

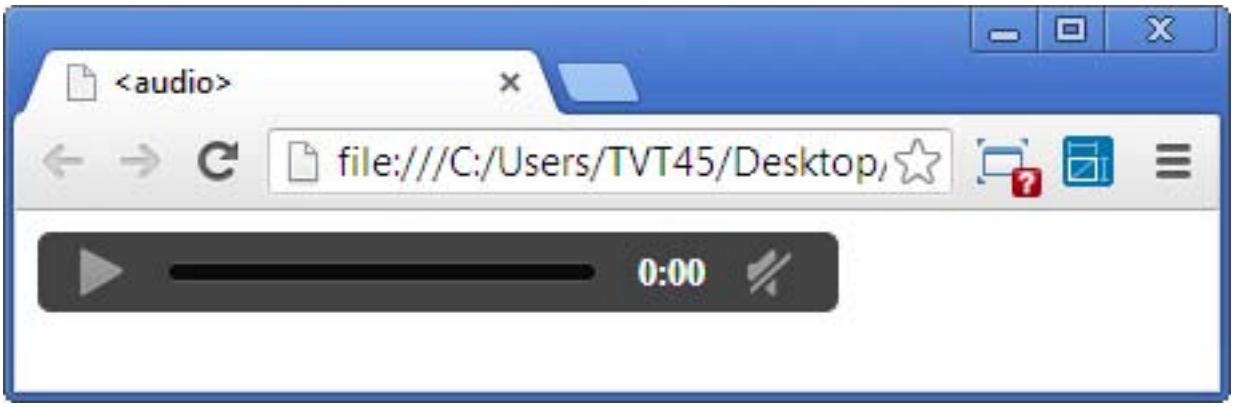
<audio> tag

- Today, most audios are shown through a plug-in (like flash). However, different browsers may have different plug-ins.
- HTML5 defines a new element which specifies a standard way to embed a audio on a web page: the <audio> element.
- The sub elements of <audio> tag are:
  - <source> : Defines the source of the audio.

Example of <audio> tag:

```
<!DOCTYPE html>
<html>
        <head>
                <title> <audio> </title>
        </head>
        <body>
                <audio controls>
                        <source src="audio1.mp3">
                </audio>
        </body>
</html>
```

Output:

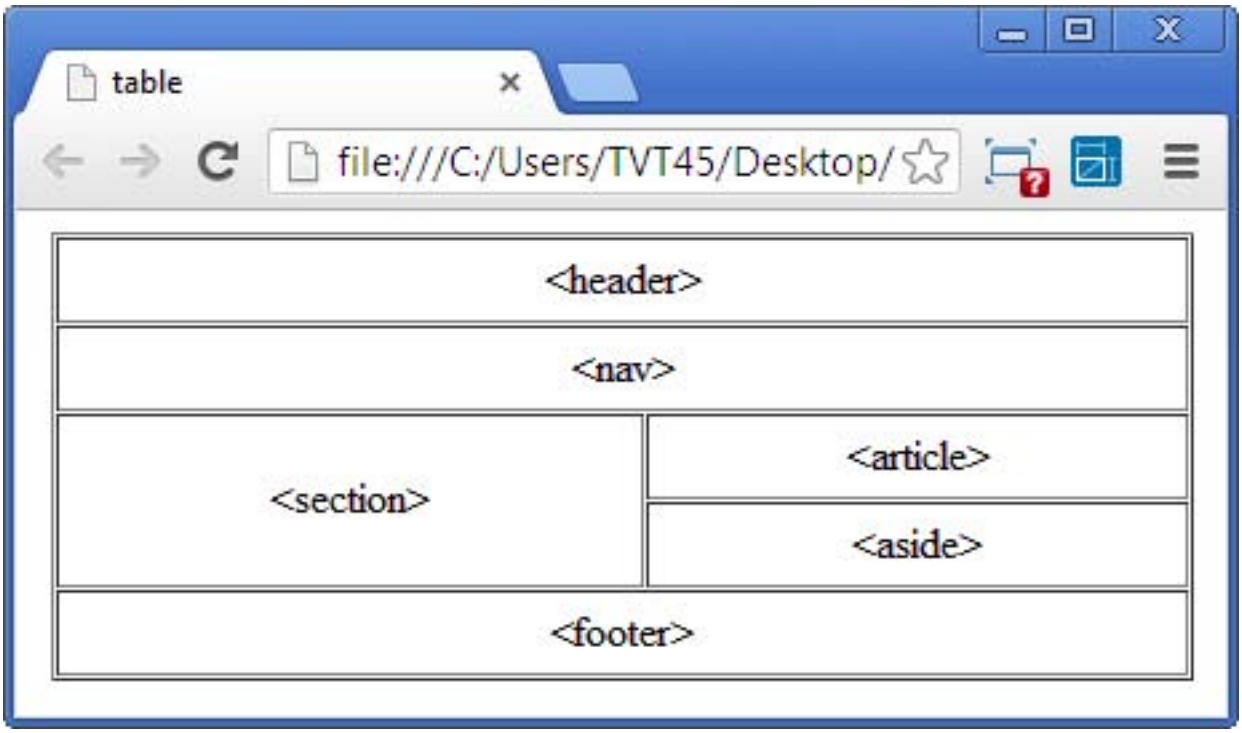


- The attribute of the <audio> tag are:
  - Control: Displays the controls like: PAUSE, PLAY and VOLUME.
- The attribute of the <source> tag are:
  - Src: Defines the source of the audio file.
  - Type: Defines the MIME type of the audio like audio/mpeg or audio/ogg or audio/wav.


HTML5
POWERED
<!DOCTYPE>
<optgroup>
<caption>
<base>
<footer>
<article>
<abbr>
<kbd>
<audio>
<figcaption>
<area>
<eventsource>
<figure>
<summary>
<input>
<progress>
<embed>
<aside>
<mark>
<meta>
<fieldset
<style>
<meter>
<hgroup>
<thead>
<legend>
<param>
<menu>
<body>
<option>
<label>
<script>
<select>
<iframe>
<time>

HTML 5 SEMANTICS and HTML5 FORMS

- HTML5 has introduced many new semantics to easily develop the format of the webpage.
- Semantics define the meaning of the block.
- The new elements that are defined as the semantics are:
  - <header> : The <header> tag is used to define the header of the webpage. It generally contains the logo or the title that defines the purpose of the webpage.
  - <footer> : The <footer> tag is used to define the footer of the webpage. It generally contains the information about the author, the copyright conditions, links to terms of use, contact information, etc.
  - <nav> : The <nav> element defines a set of navigation links. The <nav> element is intended for large blocks of navigation links. However, not all links in a document should be inside a <nav> element!
  - <section> : The <section> element defines the title of the content written in the page.
  - <article> : The <article> element is the actual body of the webpage. It contains the actual content for which the webpage is made. It contains data, pictures, links etc.
  - <aside> : The <aside> element defines some content aside from the content it is placed in (like a sidebar). It is generally used for vertical navigation, news, updates etc.
  - <figcaption> : The <figcaption> defines a caption for the figures.
  - <figure> : The <figure> element is used to define the figure and diagrams in the content.



Example For new Semantic Elements in HTML5:

```
<!DOCTYPE html>
<html>
	<head>
		<title> Semantic Elements in HTML5</title>
	</head>
	<body>
		<header align="center">
			<h2>  HELLO EVERYONE!!! GOOD MORNING </h2>
		</header>
		<nav align="center">
			<a href="#"> Home </a>
			<a href="#"> About Us </a>
			<a href="#"> Gallery </a>
			<a href="#"> Contact Us </a>
		</nav>
		<section align="center">
			<h3> Welcome to The World of HTML5 </h3>
		</section>
		<article align="center">
			<p> HTML5 is just an extension of HTML. It is not a new language to develop a web
page, It is based on the old HTML language only with some more
features.</p>
			<p> It supports features like graphics, multimedia, some new markups and API'S.

		</article>
		<aside  align="right">
			HTML5 - Wrong Notions                
<br>
HTML5 - Introduction                 <br>
```

```
HTML5 - Graphics Support                 <br>
                    HTML5 - Multimedia                 <br>
                    HTML5 - Semantics                 <br>


                </aside>
                <footer align="center"> Copyright Reserved.            
   - Developed By Monika Jain. </Footer>
        </body>
</html>
```

Output:

HTML5 Form Semantics: New Form Elements

- In HTML5 3 new Form elements are defined. They are:
  - <datalist> : The datalist element is used to define a predefined list of option for a specific input. The datalist element is hooked up to an input element using the list attribute on the input element.

Example For <datalist>:

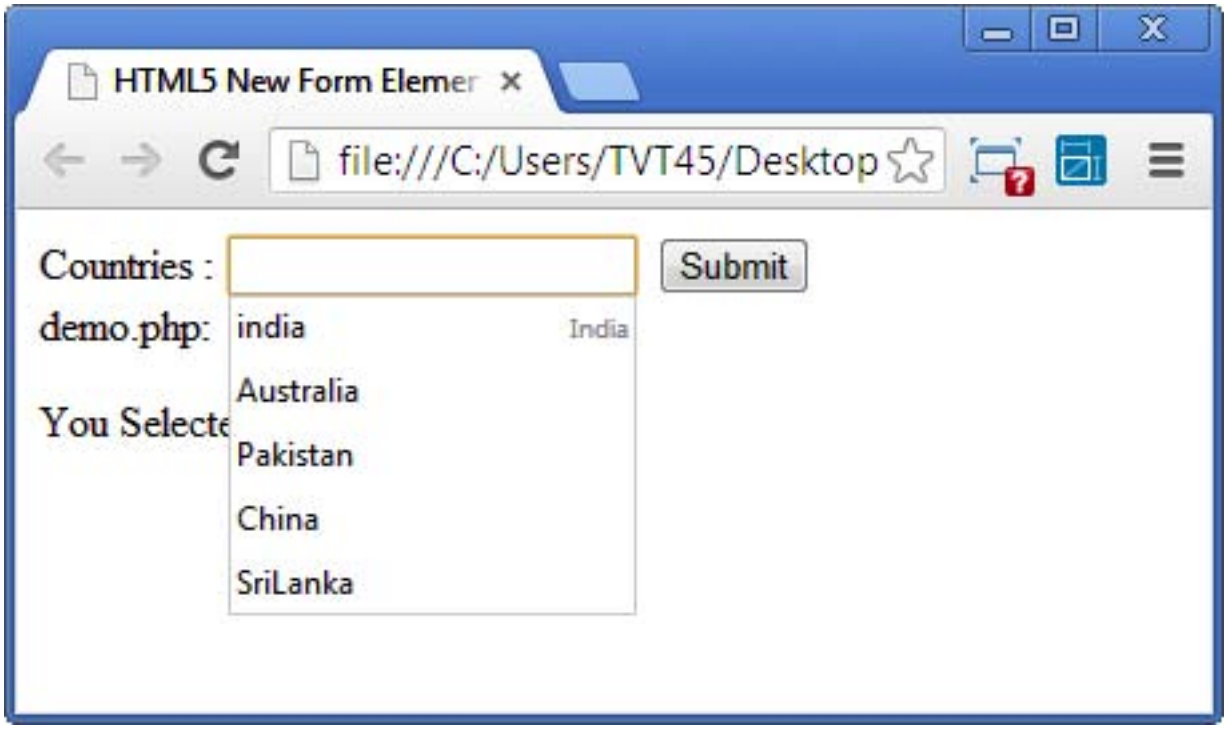
datalist.php:

```
<!DOCTYPE html>
<html>
        <head>
                <title> HTML5 New Form Elements </title>
        </head>
        <body>
                <form action="demo.php" method="post">
                        Countries : <input  list="list1" name="countries1">
                        <datalist id="list1">
                                <option value="india"> India </option>
                                <option value="Australia"> Australia </option>
                                <option value="Pakistan"> Pakistan </option>
                                <option value="China" > China </option>
                                <option value="SriLanka"> SriLanka </option>
                        </datalist>
                        <input type="submit" value="Submit" />
                </form>
        </body>
</html>
```

demo.php:

```
<html>
<body>
                <p> You Selected <?php echo $_POST["countries1"]; ?> </p>
</body>
</html>
```

Output:



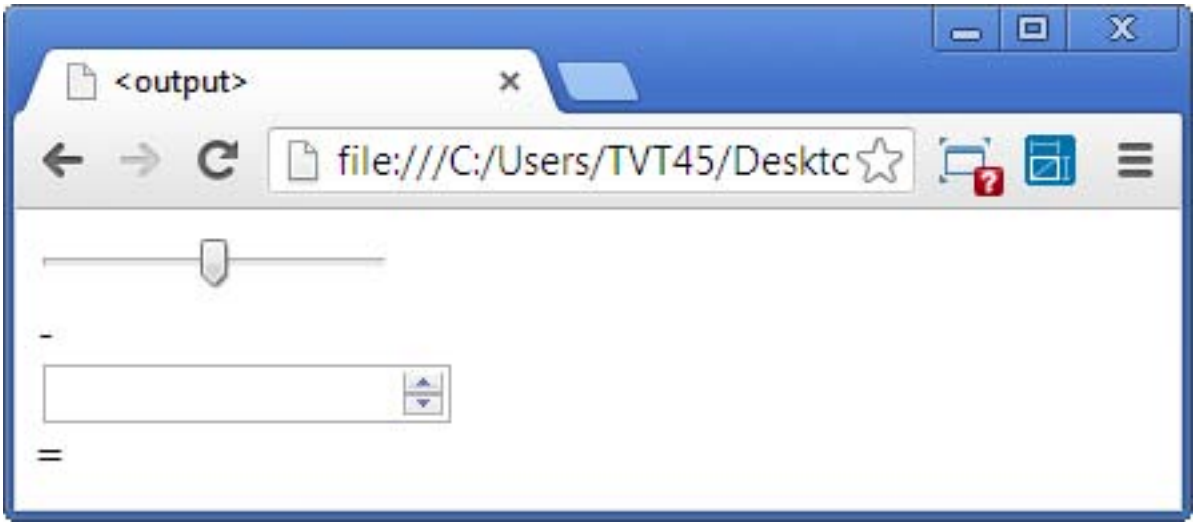
On Selecting China and Submitting the Form:

- <keygen> : It represents a key pair generator control. It generates two keys when the form is submitted: public key and private key. It is used for secure transactions.
- <output>: It is used to represent the output of a certain mathematical calculation. It uses javascript for it.

Example for <output>

```
<!DOCTYPE html>
<html>
        <head>
                <title> <output> </title>
        </head>
        <body>
                <form oninput="x.value=parseInt(a.value)-parseInt(b.value)">
                        <input type="range" id="a"> <br/>-<br/>
                        <input type="number" id="b" > <br/>= <br/>
                        <output name="x" for="a b"></output>
                </form>
</body>
</html>
```

Output:



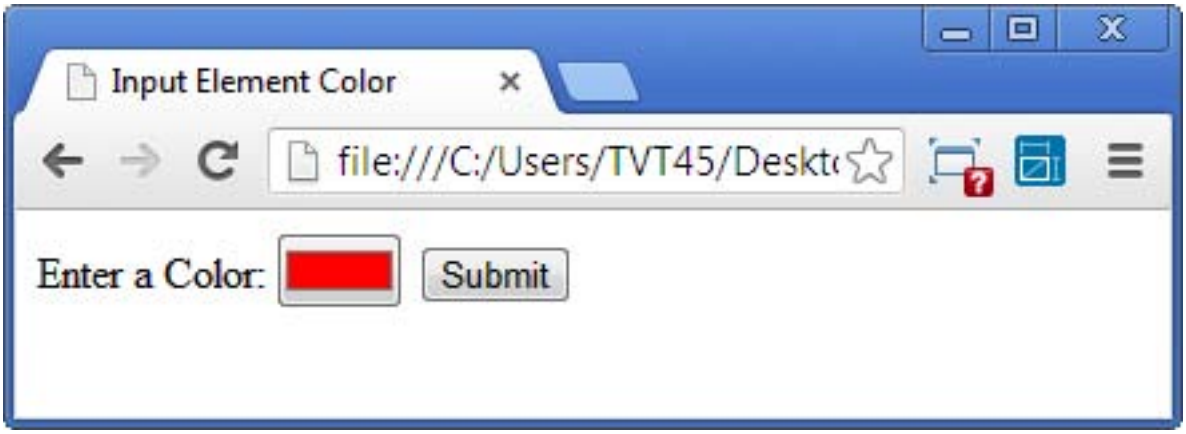
HTML5 Form Semantics: New Input types

- HTML5 allows new input types for forms. These input types add features for better control and validation.
- The new input types are:
  - **color –** If a browser supports this input type, the intention is that clicking in the textfield will result in a color chooser popping up
    - **Syntax:** <input type="color" name="name" />
    - **Attributes :**
      - Value: It defines the initial value. This color will be shown in the textfield and also when the color chooser opens it selects that color only by default.
    - **Supported in Browsers: CHROME and OPERA.**

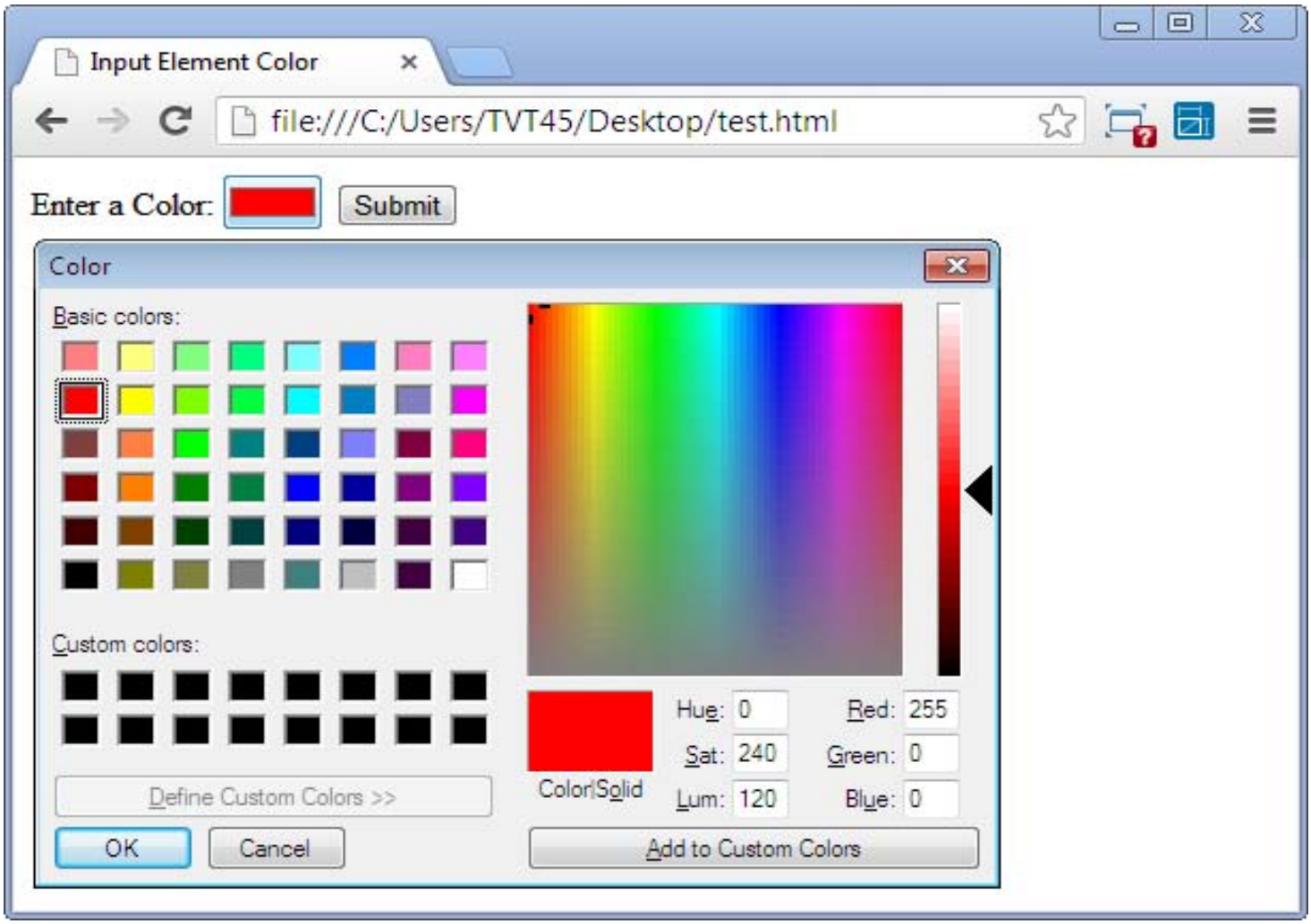
Example For <input type="color">

```
<!DOCTYPE html>
<html>
	<head>
		<title> Input Element Color </title>
	</head>
	<body>
		<form action="inputdemo.php" method="post">
			Enter a Color: <input type="color" name="color" value="#FF0000" />
			<input type="submit" value="Submit" />
		</form>
	</body>
</html>
```

Output:



On Clicking the Red Color Text field a Color Browser pops up with red color selected as default selection:

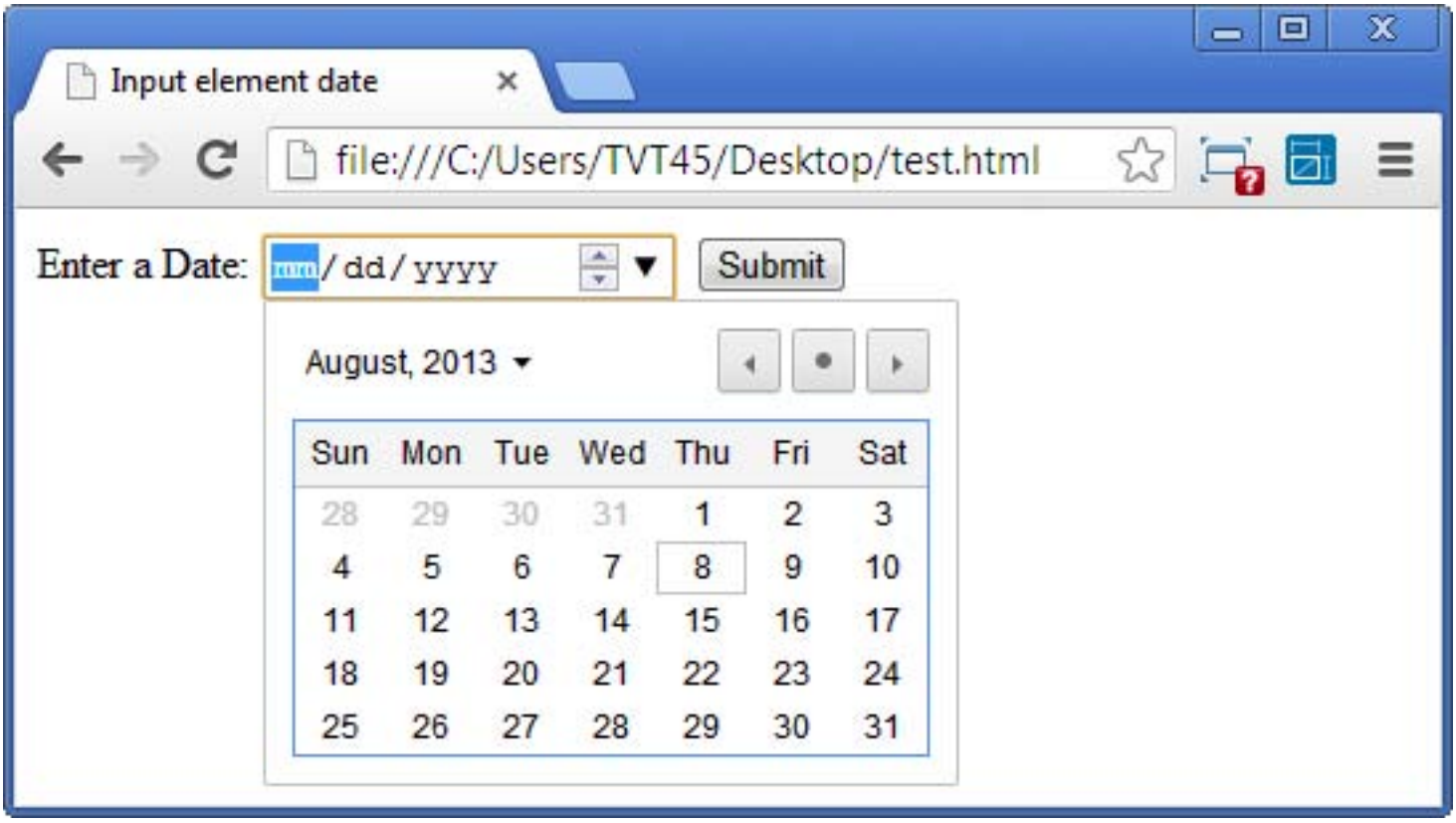


- **date –** This works the same as the datepicker. When you click the textfield a calendar is displayed and allows you to select a date from it.
  - **Syntax:** <input type="date" name="name" />
  - **Attributes :**
    - Value: The initial value that is by default displayed in the text field.
    - Step: The step size in days. By default it is 1.
    - Min , Max : The minimum and maximum date that can be selected.
  - **Supported in Browsers: CHROME, SAFARI and OPERA.**

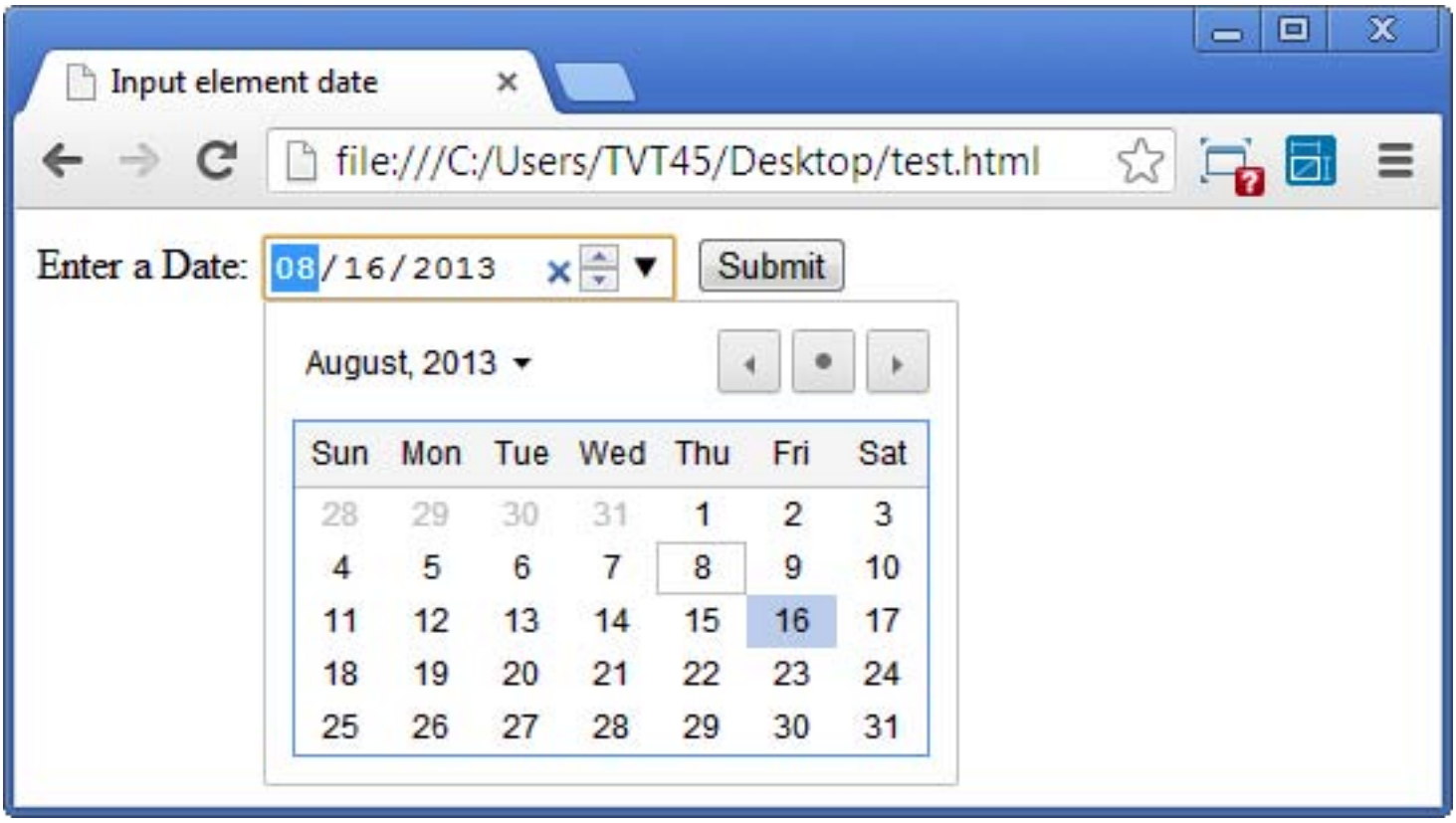
Example for <input type="date" />:

```
<!DOCTYPE html>
<html>
    <head>
        <title> Input element date </title>
    </head>
    <body>
        <form action="inputdemo.php" method="post">
            Enter a Date: <input type="date" name="date" />
            <input type="submit" value="Submit" />
        </form>
    </body>
</html>
```

Output:



On Double Clicking on 16th date the value is displayed as follows in the text field:



- **datetime:** This element allows the user to select date and time.
  - **Syntax:** <input type="datetime" name="name" />
  - **Attributes :**
    - **Value=** Sets the initial value.
  - **Supported in Browsers: SAFARI and OPERA.**
- **email –** This element allows you to enter a valid email address.
  - **Syntax:** <input type="email" name="name" />
  - **Attributes :**
    - Value: It is used to define an initial value.
    - List: Defines the id of datalist to be displayed.
  - **Supported in Browsers: IE, FIREFOX, CHROME and OPERA.**

Example for <input type="email" />

email.html

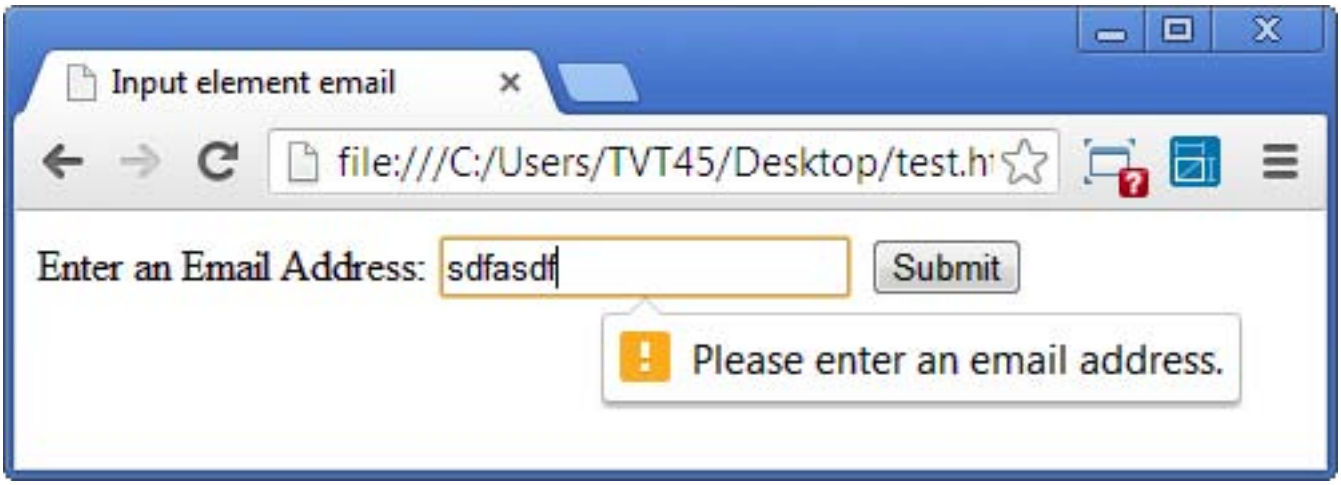
```
<!DOCTYPE html>
<html>
	<head>
		<title> Input element email </title>
	</head>
	<body>
		<form action="inputdemo.php" method="post">
			Enter an Email Address: <input type="email" name="name" />
			<input type="submit" value="Submit" />
		</form>
	</body>
</html>
```

inputdemo.php

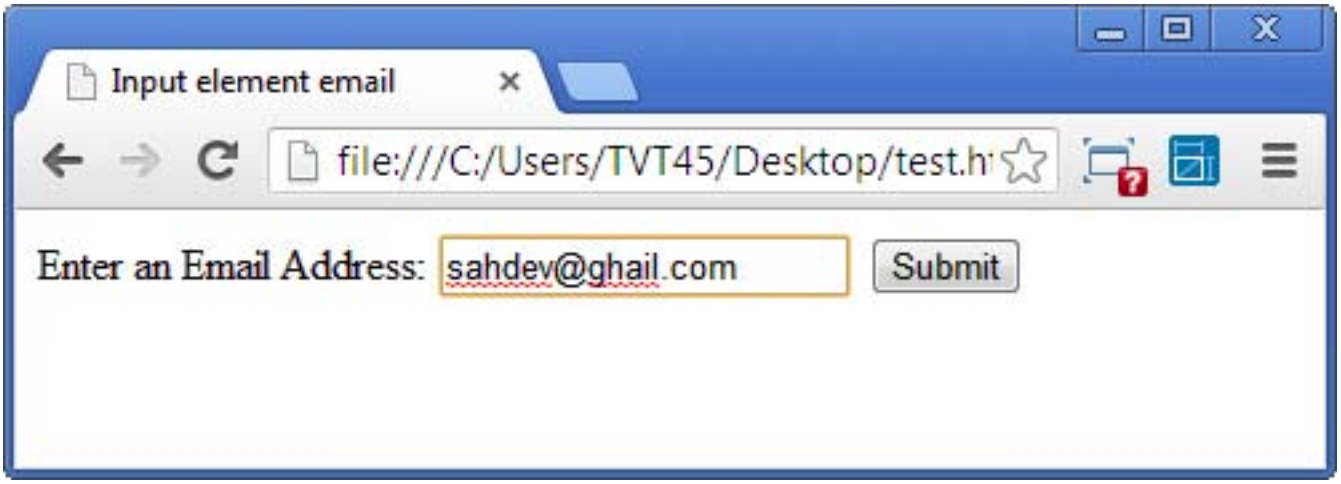
```
<html>
	<body>
		<?php
			$email = $_POST["email"];
			echo "You have entered the email address:" . $email;
		?>
	</body>
</html>
```

Output:

On entering a Wrong Email Address and hitting Submit the output is as follows:



On entering a correct Email Address:

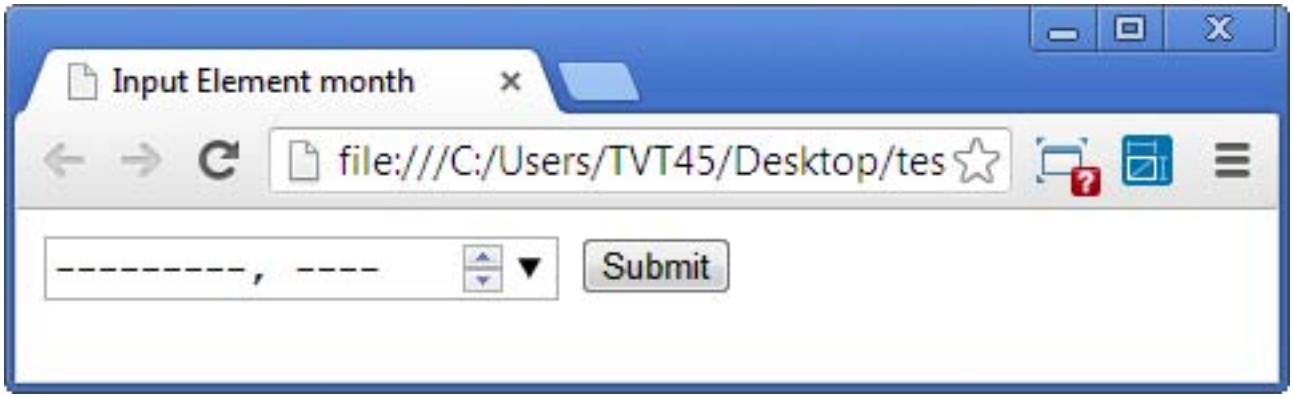


- **month –** This element allows the user to select a month and year.
  - **Syntax:** <input type="month" name="name" />
  - **Attributes :**
    - **Value:** It defines the initial value.
  - **Supported in Browsers: CHROME, SAFARI and OPERA.**

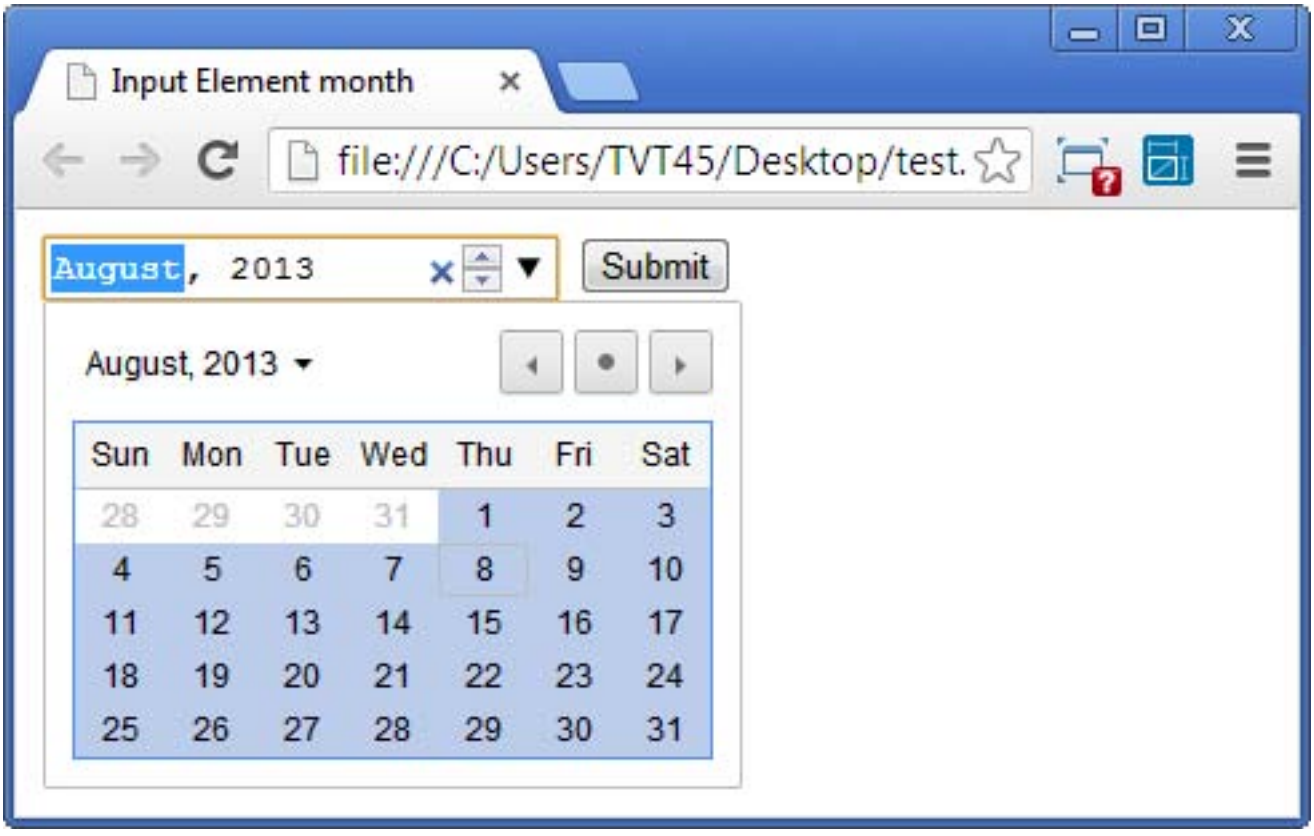
Example for <input type="month" />:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Element month </title>
        </head>
        <body>
                <form action="inputdemo.php" method="post">
                        <input type="month" name="month" />
                        <input type="submit" value="Submit" />
                </form>
        </body>
</html>
```

Output:



Now in this first you have to select the month or year field and then click the up or down arrow: For month:



- **number –** This element lets you select a number. It is also known as spinner.
  - **Syntax:** <input type="number" name="name" />
  - **Attributes :**

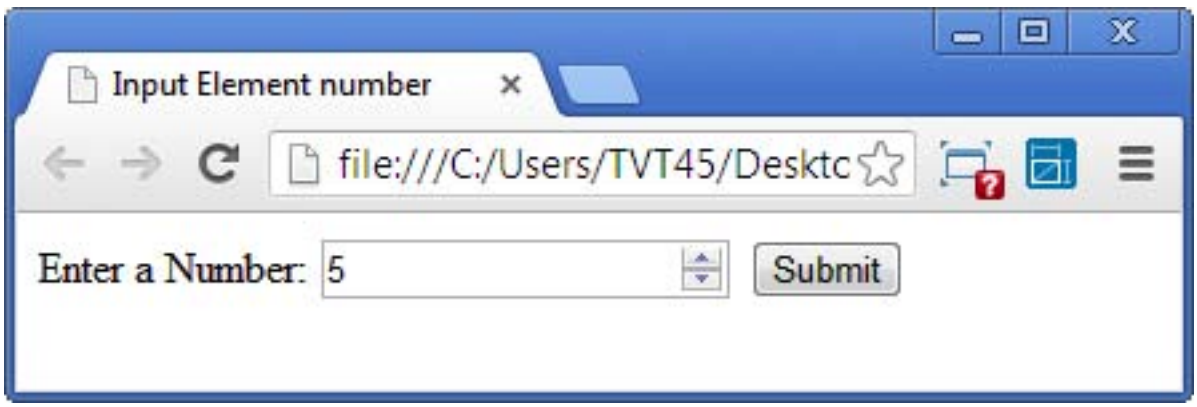
- **Value:** It defines the initial value. If this field is not set the field is defined blank.
- **Step:** It defines the step size. The difference in the selected number is number displayed after clicking up and down arrow is called step. By default it is 1.
- min and max: The minimum and maximum value that can be selected on clicking the up and down arrows.

- **Supported in Browsers: CHROME, SAFARI and OPERA.**

Example for <input type="number" />

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Element number </title>
        </head>
        <body>
                <form action="inputdemo.php" method="post">
                        Enter a Number: <input type="number" value="5" step="2" name="number"/>
                        <input type="submit" value="Submit" />
                </form>
        </body>
</html>
```

Output:



- **range –** This element allows the user to select a value between the given range.
  - **Syntax:** <input type="range" name="name" />
  - **Attributes :**
    - **Value:** It defines the initial value. Also it is always halfway between the min and max.
    - **Step:** It change in the number on moving the slider is known as step. It is by default 1.
    - **min, max –** Defines the minimum and maximum value of the range.
  - **Supported in Browsers: IE, CHROME, SAFARI and OPERA.**

Example for <input type="range" />:

range.html:

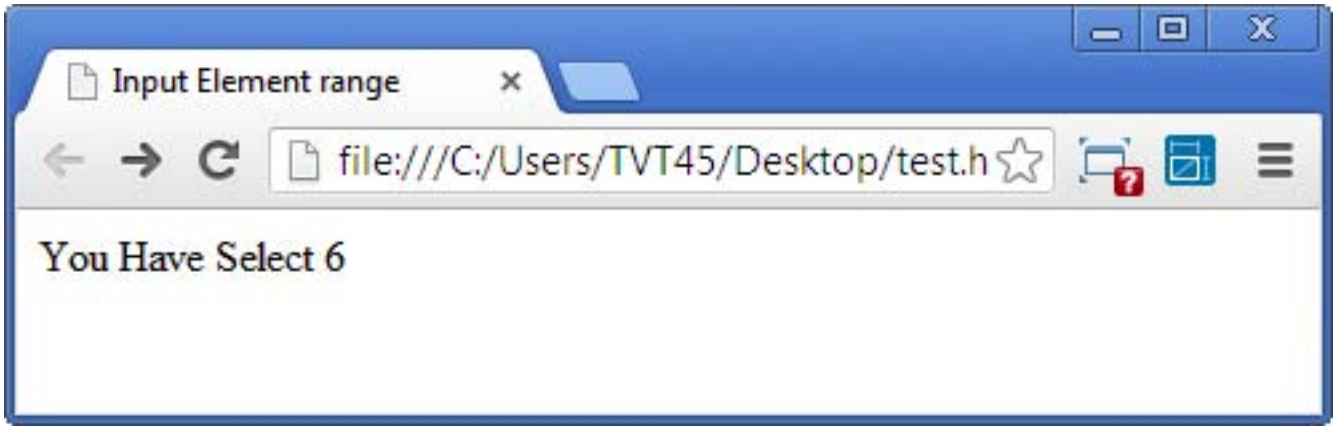
```
<!DOCTYPE html>
<html>
	<head>
		<title> Input Element range </title>
	</head>
	<body>
		<form action="inputdemo1.php" method="post">
			Enter a Number in the Range (0 to 10): <input type="range" value="5" step="2"
name="range" min="0" max="10"/>
			<input type="submit" value="Submit" />
		</form>
	</body>
</html>
```

inputdemo1.php:

```
<html>
	<body>
		<?php
			$n = $_POST["range"];
			echo "You have selected" . "" . $n;
		?>
	</body>
</html>
```

Output:

On hitting Submit:

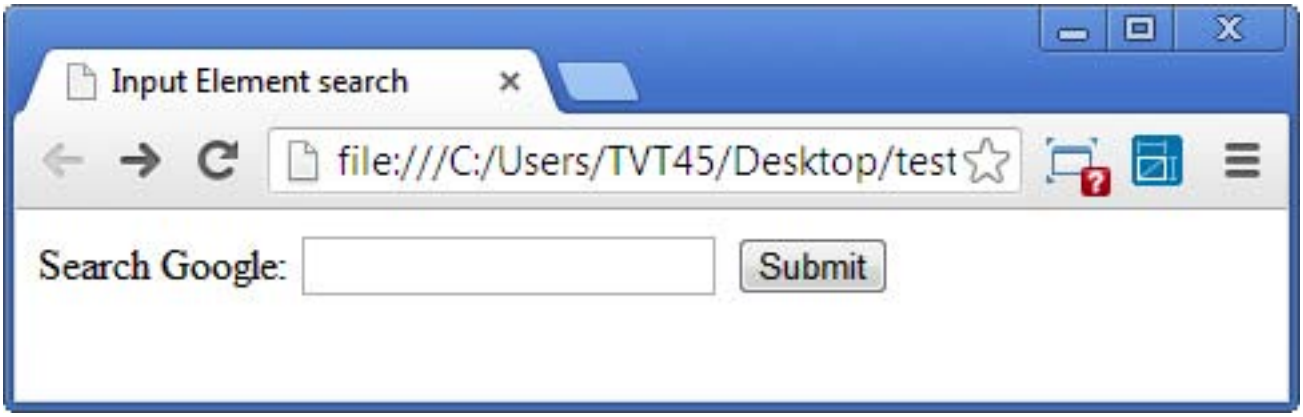


- **search –** This element allows you to enter a string that can be used to search anything in the database.
  - **Syntax:** <input type="search" name="name" />
  - **Attributes :**
    - **Value:** Sets the initial value.
  - **Supported in Browsers: CHROME and SAFARI.**

Example for <input type="search" />

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Element search </title>
        </head>
        <body>
                <form action="inputdemo3.php" method="post">
                        Search Google: <input type="search" name="search" />
                        <input type="submit" value="Submit" />
                </form>
        </body>
</html>
```

Output:



- **tel –** This element allows the user to enter a valid telephone no.
  - **Syntax:** <input type="tel" name="name" />
  - **Attributes :**
    - **Value:** Sets an initial value.
  - **Is not supported in any of the BROWSERS.**

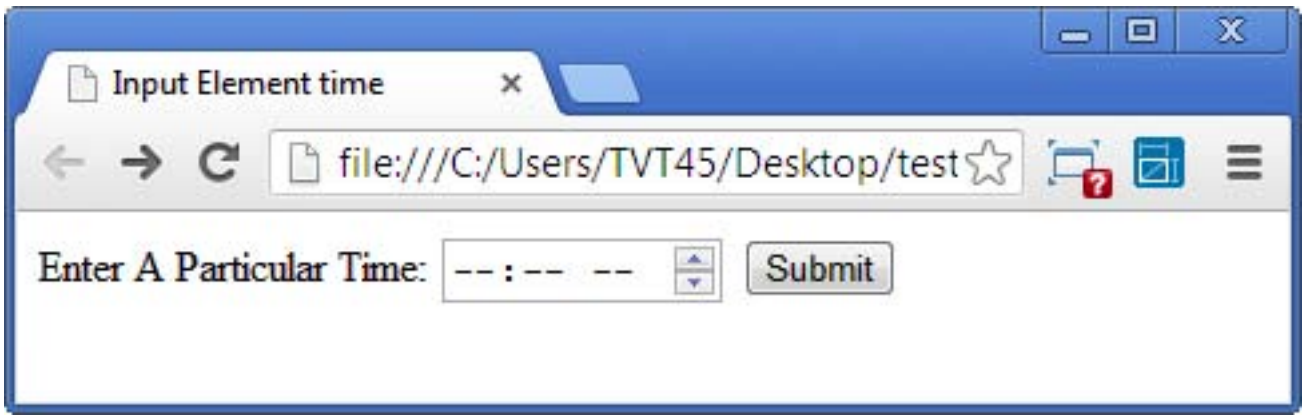
I m sorry friends I won't be able to give an example for <input type="tel" as I won't be able to show you the output because this input type is not supported in any browser.

- **time –** This element allows the user to select time.
  - **Syntax:** <input type="time" name="name" />
  - **Attributes :**
    - **Value:** Sets the initial value.
  - **Supported in CHROME, SAFARI and OPERA.**

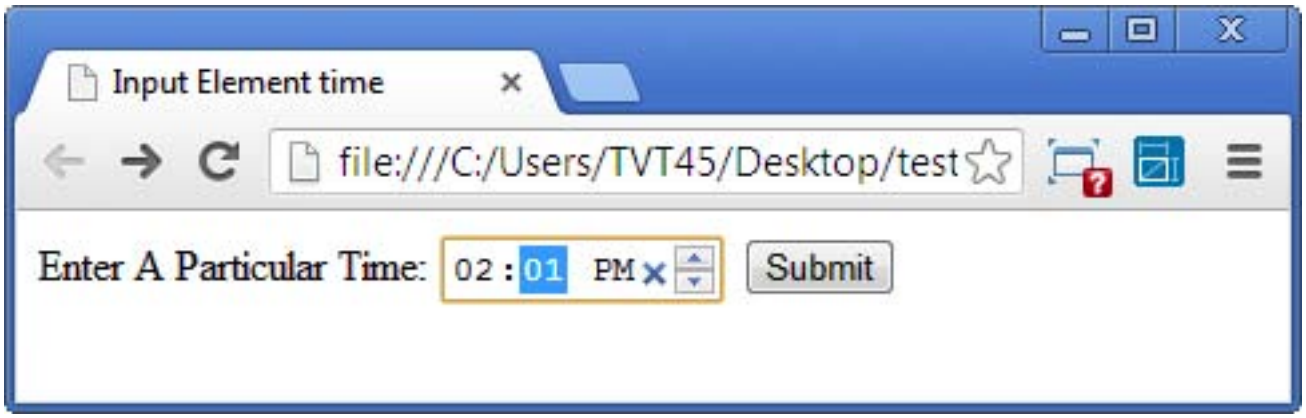
Example for <input type="time" />

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Element time </title>
        </head>
        <body>
                <form action="inputdemo.php" method="post">
                        Enter A Particular Time: <input type="time" name="time" />
                        <input type="submit" value="Submit" />
                </form>
        </body>
</html>
```

Output:



Now on selecting a particular value:



- **url –** This element allows the user to enter a valid url address.
  - **Syntax:** <input type="url" name="name" />
  - **Attributes :**
    - **Value:** Sets the initial value.
  - **Supported in IE, CHROME, FIREFOX and OPERA.**

Example for <input type="url" />

url.html

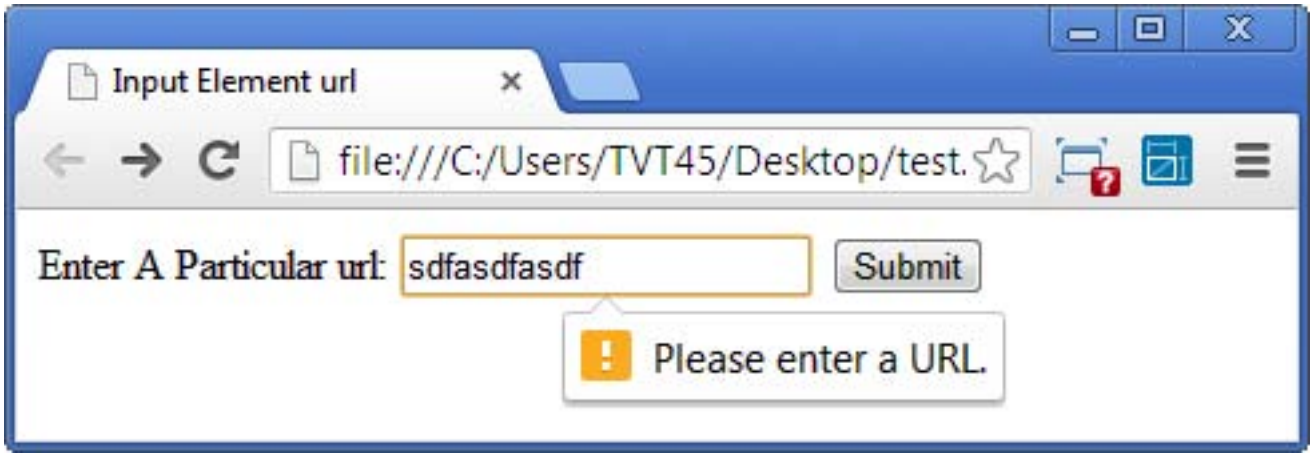
```
<!DOCTYPE html>
<html>
	<head>
		<title> Input Element url </title>
	</head>
	<body>
		<form action="inputdemo2.php" method="post">
			Enter A Particular url: <input type="url" name="url" />
			<input type="submit" value="Submit" />
		</form>
	</body>
</html>
```

inputdemo2.php

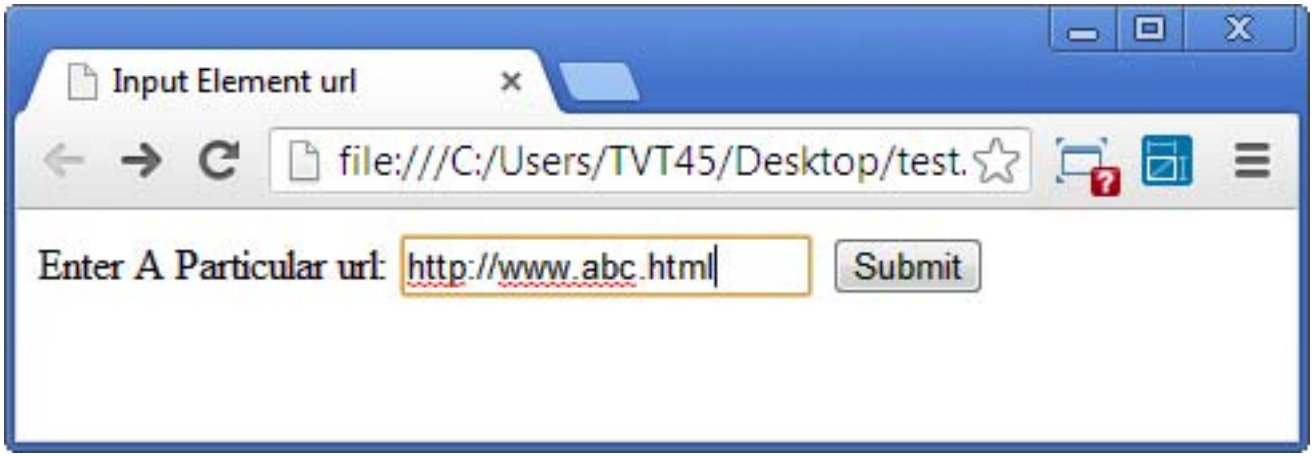
```
<html>
	<body>
		<?php
			$url = $_POST["url"];
			echo "You have selected" . " " . $url;
		?>
	</body>
</html>
```

Output:

On entering a Wrong URL and hitting Submit the output is as follows:



On entering a Wrong URL and hitting Submit the output is as follows:



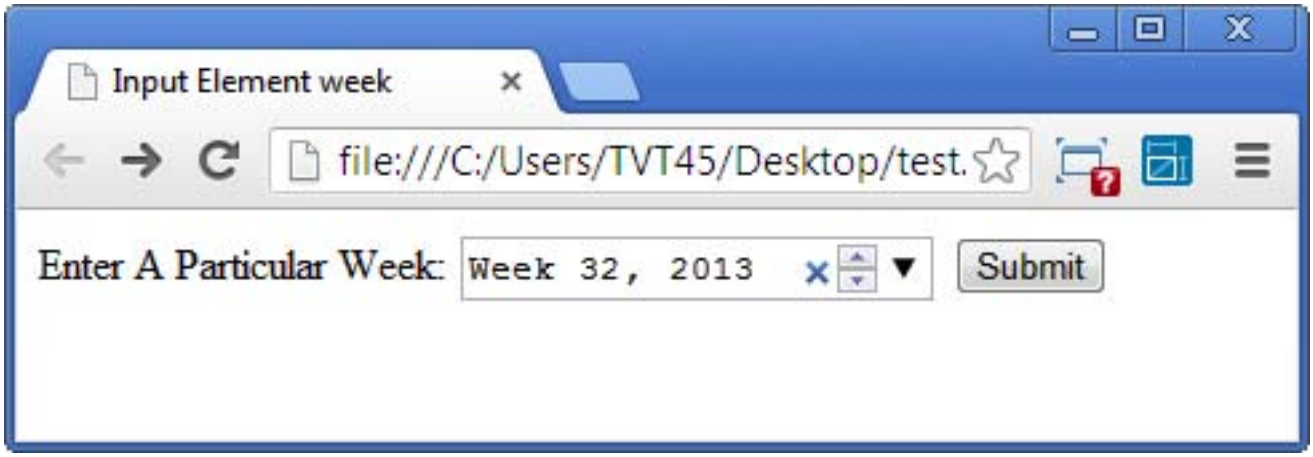
Now on hitting Submit:

- **week –** This element allows the user to select a week and a year.
  - **Syntax:** <input type="week" name="name" />
  - **Attributes :**
    - **Value:** Sets the initial value.
  - **Supported in CHROME, SAFARI and OPERA.**

Example for <input type="week" />

```
<!DOCTYPE html>
<html>
	<head>
		<title> Input Element week </title>
	</head>
	<body>
		<form action="inputdemo2.php" method="post">
			Enter A Particular Week: <input type="week" name="week" />
			<input type="submit" value="Submit" />
		</form>
	</body>
</html>
```

Output:



HTML5 Form Semantics: New Form Attributes

- HTML5 has defined many new attributes for <form> tag and <input> tag.

  The attributes defined for <form> tag are:

  - autocomplete: When the autocomplete attribute is on the value of entered by the user will be automatically completed by the values entered by the user before. If it is off the user will have to enter the whole value.
    - Syntax: <form autocomplete="off/on"> / <input autocomplete="off/on" />
    - autocomplete only works with input type text, search, url, tel, email, password, datepickers, range, and color.
    - **Supported in Browsers: IE, FireFox, Chrome and Safari**

Example for attribute "autocomplete":

autocomplete.html

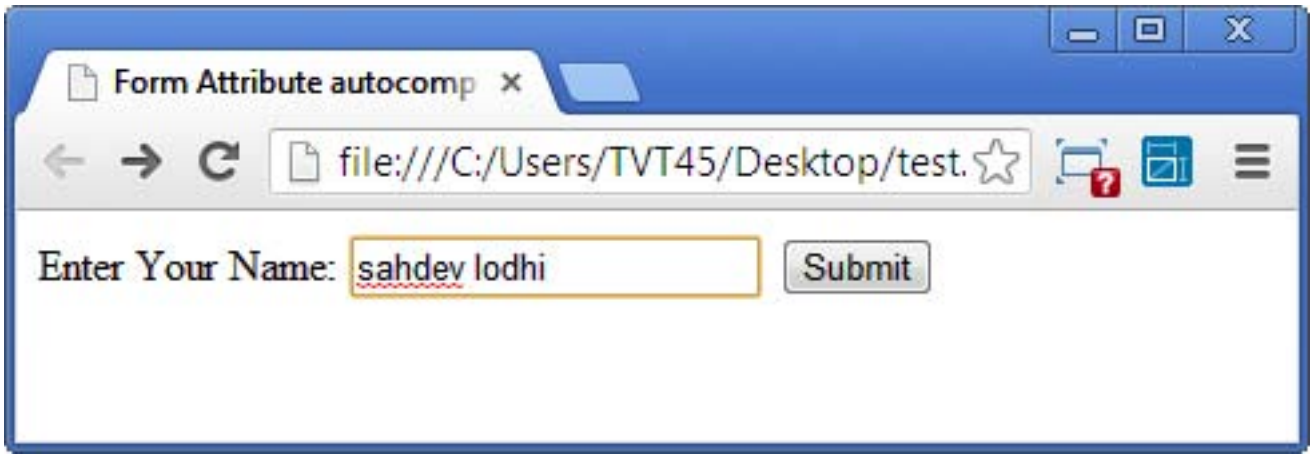
```
<!DOCTYPE html>
<html>
	<head>
		<title> Form Attribute autocomplete </title>
	</head>
	<body>
		<form action="attributedemo.php" method="post" autocomplete="off">
			Enter Your Name: <input type="text" name="name" />
			<input type="submit" value="Submit" />
		</form>
	</body>
</html>
```

attributedemo.php

```
<html>
	<body>
		<?php
			$name = $_POST["name"];
			echo "You Entered Name: " . $name;
		?>
	</body>
</html>
```

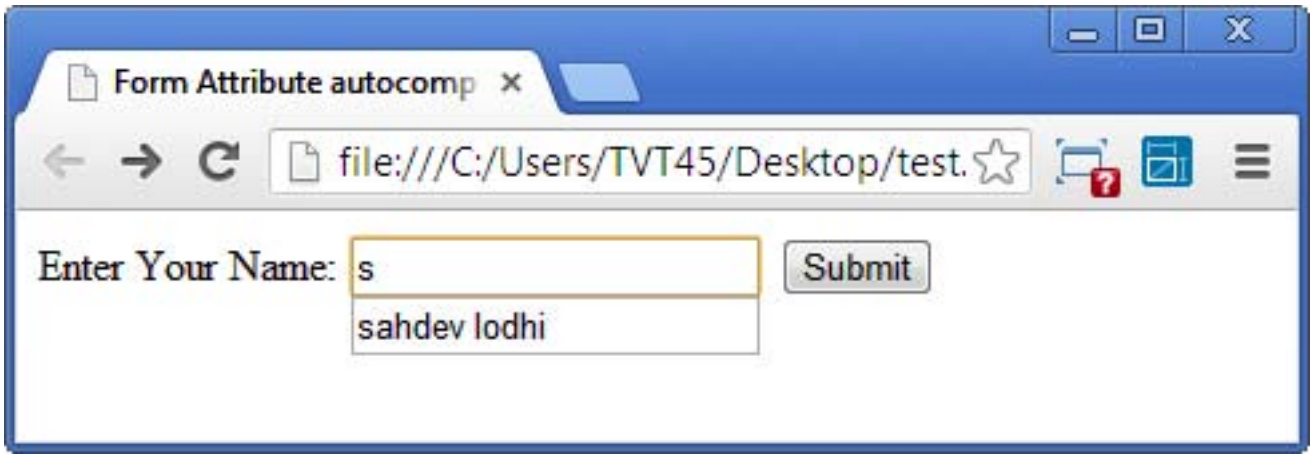
Output:

Fill the Form: autocomplete ="on"

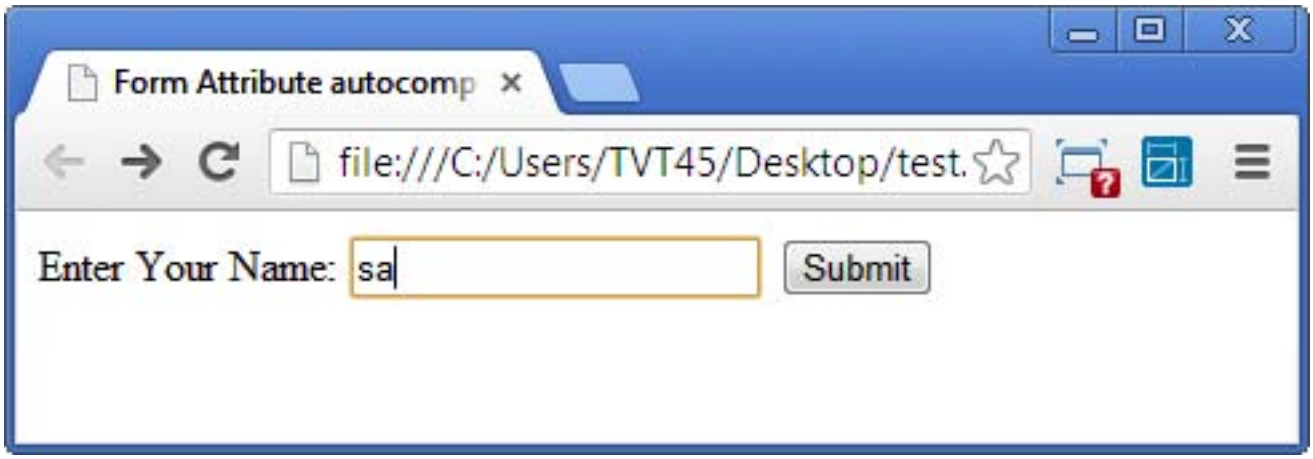


On hitting Submit:

Now on reloading and entering character 's'



Now on setting autocomplete="off": It's not showing the value I entered before.



- **novalidate:** If you specify this attribute the data entered in the form will not be validated and submitted directly.
  - Syntax: <form novalidate>
  - **Supported in Browsers: FireFox, Chrome and Opera**

Example for Attribute novalidate:

novalidate.html

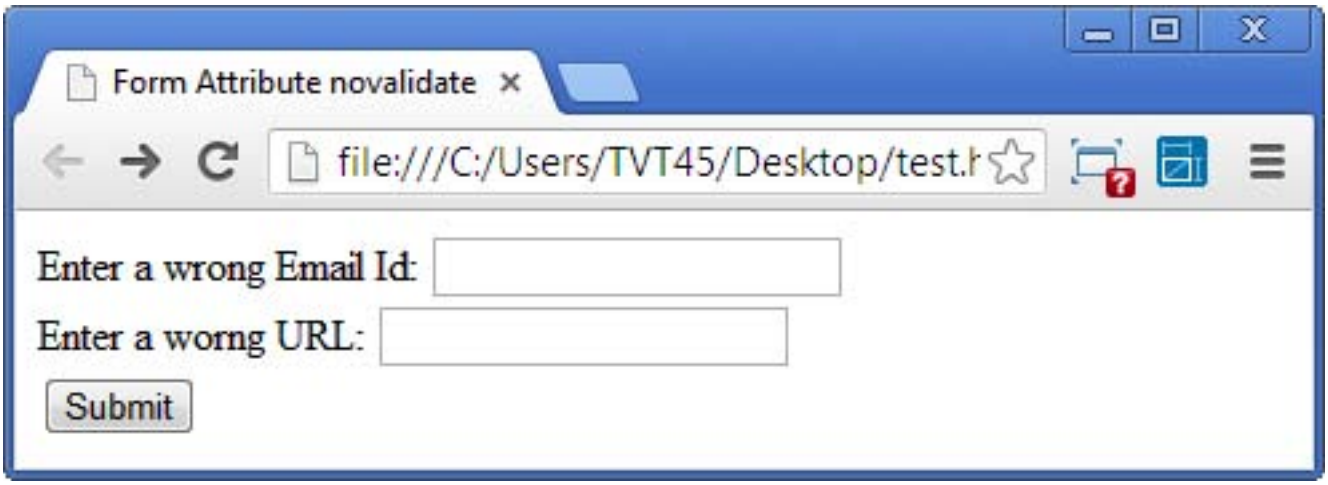
```
<!DOCTYPE html>
<html>
	<head>
		<title> Form Attribute novalidate </title>
	</head>
	<body>
		<form action="attributedemo1.php" method="post" novalidate>
			Enter a wrong Email Id: <input type="email" name="email" /><br />
			Enter a worng URL: <input type="url" name="url" /><br />
			<input type="submit" value="Submit" />
		</form>
	</body>
</html>
```

attributedemo1.php

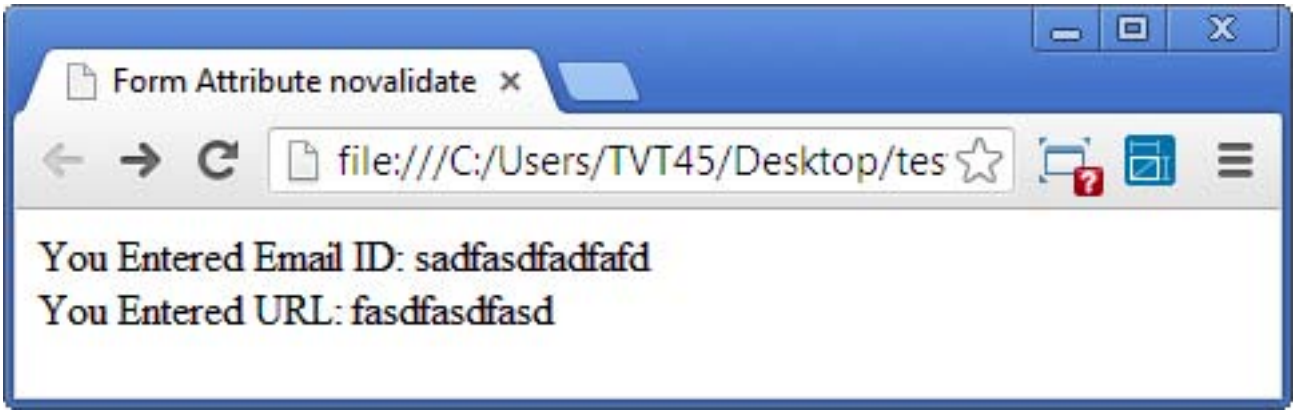
```
<html>
	<body>
		<?php
			$email = $_POST["email"];
			$url = $_POST["url"];
			echo "You Entered Email ID: " . $email . "<br>";
			echo "You Entered URL: " . $url;
		?>
	</body>
</html>
```

Output:

Fill the Form:



On hitting submit in actual the validation alert is displayed as we saw before in input elements but here the form gets submitted and the output is displayed as follows: This is all because of the novalidate attribute. It doesn't allow the element to perform validation.



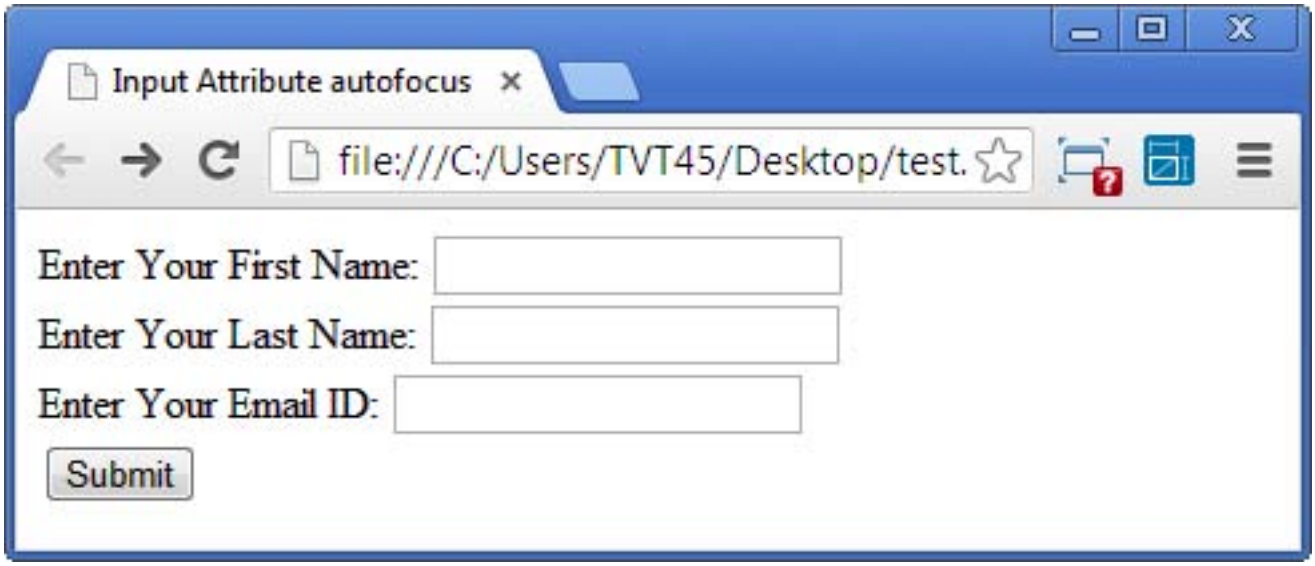
- The attributes defined for the <input> tag are:
  - **autofocus:** When you use autofocus for any input tag in the form and load the form that input will be in focus by default.
    - **Syntax -** <input type="" autofocus />
    - **Supported by all the Browsers.**

Example for Attribute "autofocus":

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Attribute autofocus </title>
        </head>
        <body>
                <form action="attributedemo.php" method="post">
                        Enter Your First Name: <input type="text" name="fn" /> <br/>
                        Enter Your Last Name: <input type="text" name="ln" /> <br/>
                        Enter Your Email ID: <input type="email" name="email" autofocus /> <br/>
                        <input type="submit" value="Submit" />
                </form>
        </body>
</html>
```

Output:

As you see we have set Email id on autofocus and when we loaded the page email id is in focus by default.



- form: This attribute is used when you want to define an input tag associated with a form outside the form tag.
  - You have to provide an id to the form tag for this so that we can uniquely associate an input tag to that form.
  - **Syntax:** <input type="" form="form_id" />
  - **Supported in Browsers: FireFox , Chrome, Safari and Opera.**

Example for Attribute "form":

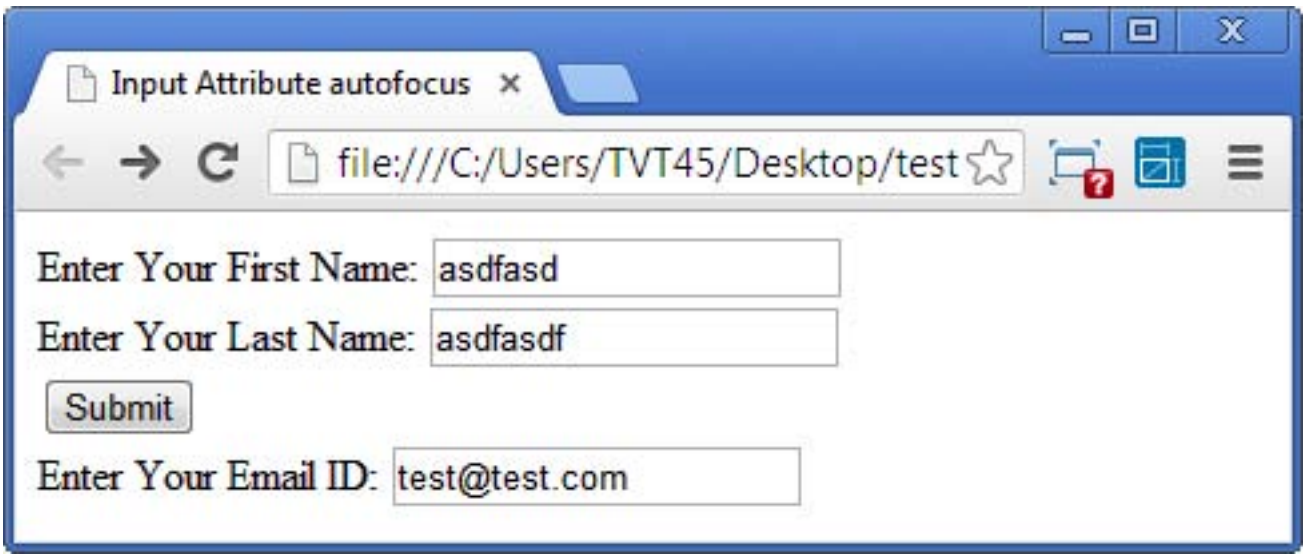
form.html:

```
<!DOCTYPE html>
<html>
	<head>
		<title> Input Attribute autofocus </title>
	</head>
	<body>
		<form action="attributedemo2.php" method="post" id="form1">
			Enter Your First Name: <input type="text" name="fn" /> <br/>
			Enter Your Last Name: <input type="text" name="ln" /> <br/>
			<input type="submit" value="Submit" />
		</form>

		Enter Your Email ID: <input type="email" name="email" form="form1" /> <br/>
	</body>
</html>
```

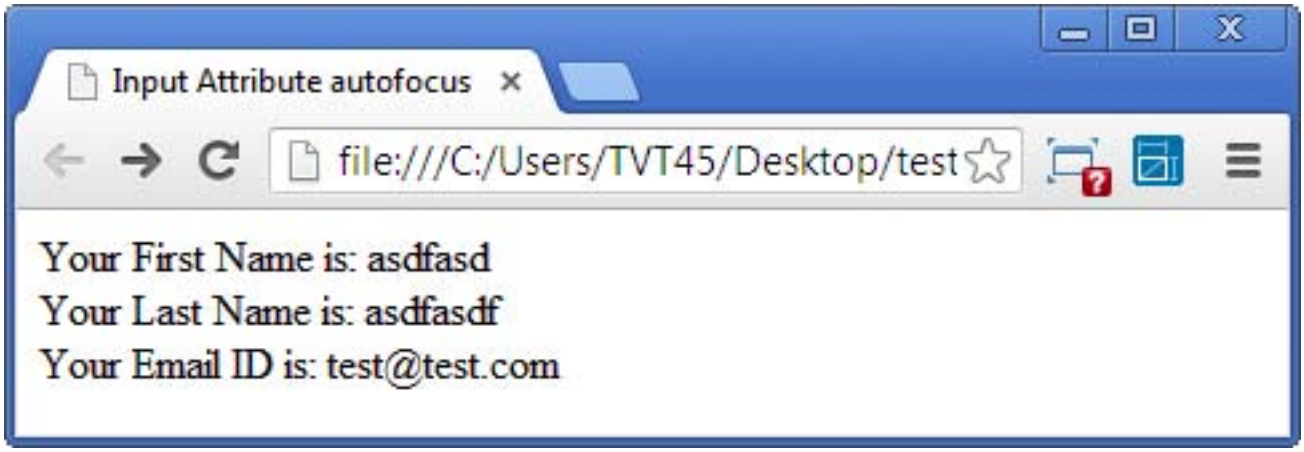
attributedemo2.php:

```
<html>
	<body>
		<?php
			$fn = $_POST["fn"];
			$ln = $_POST["ln"];
			$email = $_POST["email"];

			echo "Your First Name is: " . $fn . "<br>";
			echo "Your Last Name is: " . $ln . "<br>";
			echo "Your Email ID is: " . $email . "<br>";
		?>
	</body>
</html>
```

Output:



On hitting Submit the email-id will also be passed to the php file:



- **formaction:** 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.
  - The formaction attribute is only used with the input type image and submit.
  - Syntax: <input type="image/submit" formaction="name_of_file_where _to_submit" />

Example for Attribute "formaction":

```
<!DOCTYPE html>
<html>
	<head>
		<title> Input Attribute formaction </title>
	</head>
	<body>
		<form action="attributedemo3.php" method="post">
			Enter Your First Name: <input type="text" name="fn" /><br/>
			Enter your Last Name: <input type="text" name="ln" />
			<input type="submit" value="Submit"/>
			<input type="submit" value="Submit to AttributeDemo4.php"
formaction="attributedemo4.php"/>

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

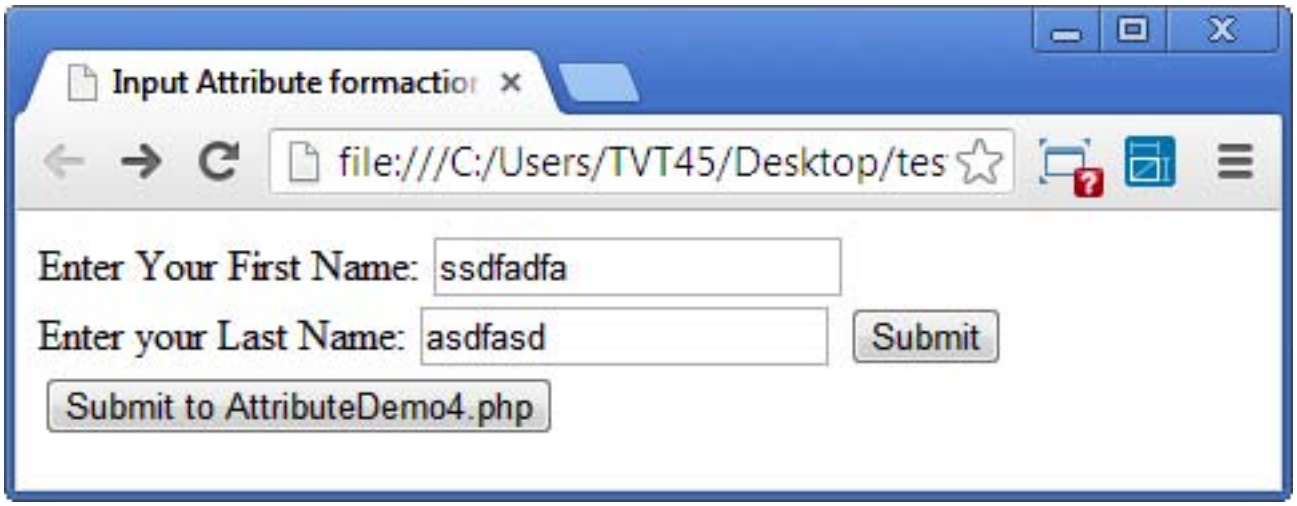
attributedemo3.php:

```
<html>
	<body>
		<?php
			$fn= $_POST["fn"];
			$ln= $_POST["ln"];
			echo "We are in AttributeDemo3.php <br>";
			echo "Your First Name is: " . $fn . "<br>";
			echo "Your Last Name is: " . $ln;
		?>
	</body>
</html>
```

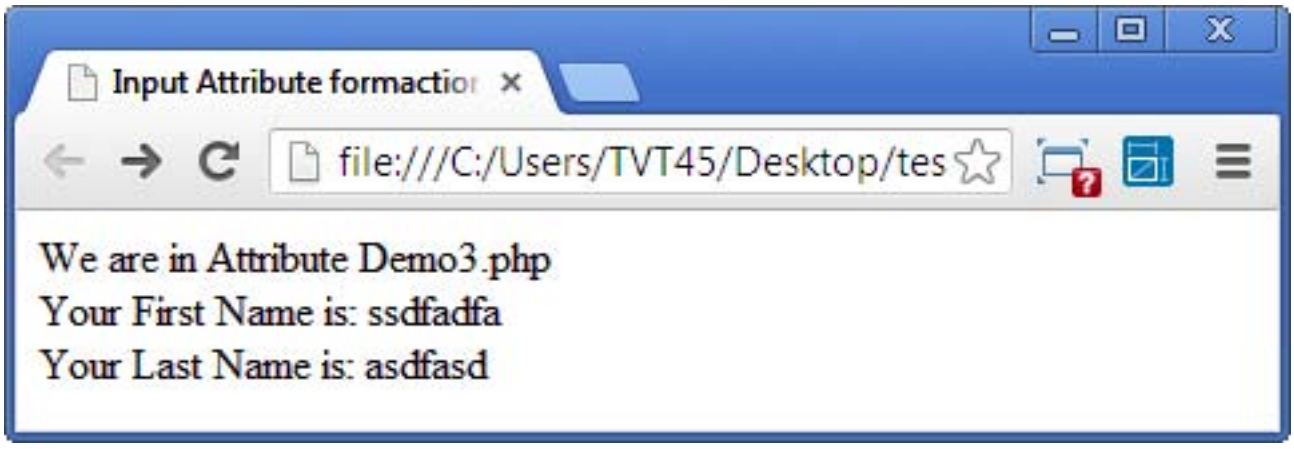
attributedemo4.php:

```
<html>
	<body>
		<?php
			$fn= $_POST["fn"];
			$ln= $_POST["ln"];
			echo "We are in AttributeDemo4.php <br>";
			echo "Your First Name is: " . $fn . "<br>";
			echo "Your Last Name is: " . $ln;
		?>
	</body>
</html>
```

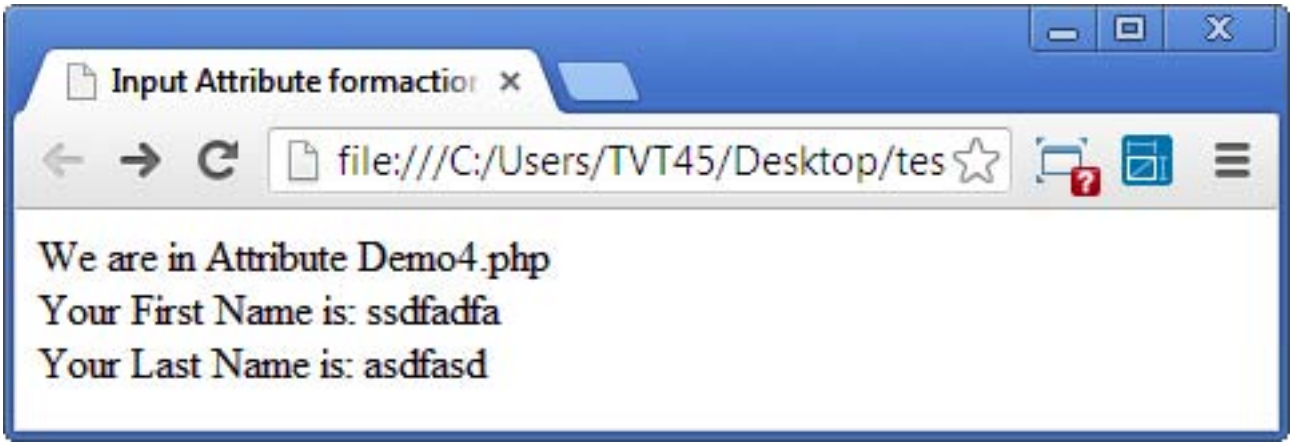
Output:



On Clicking Submit:



Now On Clicking Submit to AttibuteDemo4.php:



- **formenctype:** This attribute encodes the data posted from the form to the server.
  - The formenctype attribute overrides the enctype attribute of the <form> element.
  - This attributes can only be used with the input types: submit and image.
  - **Syntax:** <input type="submit/image" formenctype="multipart/form-data" />

Example for Attribute formenctype:

- **formmethod:** This attribute defines the HTTP method for sending the data of the form.
  - It overrides the effect of method attribute stated in the form tag.
  - This attribute can only be used with the input types: submit and image.
  - **Syntax:** <input type="submit/image" formmethod="get/post" />

Example for Attribute formmethod:

formmethod.html:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Attribute formenctype </title>
        </head>
        <body>
                <form action="attributedemo7.php" method="get">
                        Enter Your First Name: <input type="text" name="fn" /><br/>
                        Enter Your Last Name: <input type="text" name="ln" /> <br/>
                        <input type="submit" value="Submit"/>
```

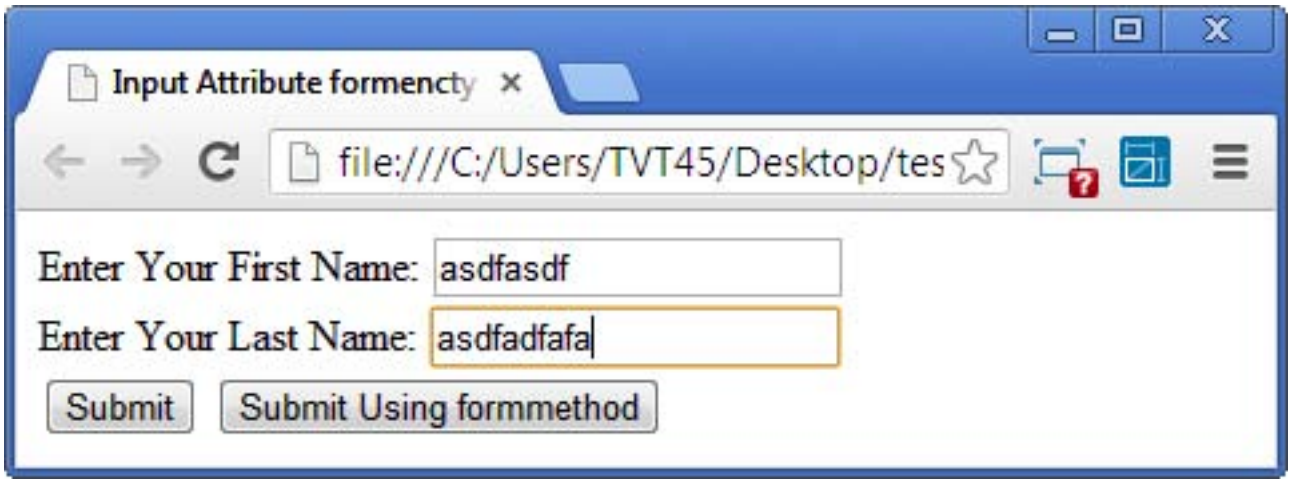
```
<input type="submit" value="Submit Using formmethod" formaction="attributedemo8.php"
formmethod="post"/>
                </form>
        </body>
</html>
```

attributedemo7.php:

```
<html>
        <body>
                <?php
                        $fn= $_GET["fn"];
                        $ln= $_GET["ln"];
                        echo "This is the data submitted using get method <br>";
                        echo "Your First Name is: " . $fn . "<br>";
                        echo "Your Last Name is: " . $ln;
                ?>
        </body>
</html>
```

attributedemo8.php:

Output:



On Clicking the Submit Button the get method specified in the form tag will be executed:

On Clicking the Submit Using formmethod Button the formmethod will be executed.
As Here We have specified the value post, the form will be submitted using post method.

- **formnovalidate:** This attribute is used when we donot want to validate the data entered in the form.
  - Works same as novalidate attribute.
  - This attribute can only be used with the input types: submit.
  - **Synatx:** <input type="submit/image" formnovalidate />

For example refer novalidate attribute. Just replace novalidate with formnovalidate and move it from form tag to <input type="submit" />

- formtarget: The formtarget attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.
  - **Values:**
    - _blank - The response is displayed in a new window or tab
    - _self - The response is displayed in the same frame (this is default)
    - _parent - The response is displayed in the parent frame
    - _top - The response is displayed in the full body of the window
    - framename - The response is displayed in a named iframe
  - This attribute can only be used with the input types: submit and image.
  - Syntax : <input type="submit/image" formtarget="_blank | _self | _parent | _top | framename" />

Example for attribute formtarget:

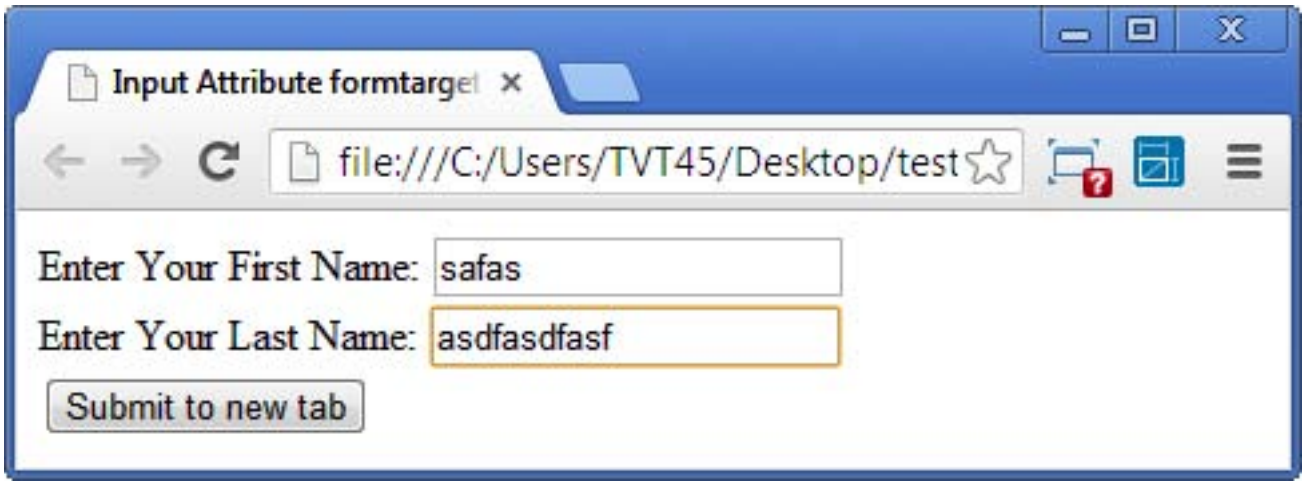
formtarget.html:

```
<!DOCTYPE html>
<html>
	<head>
		<title> Input Attribute formtarget </title>
	</head>
	<body>
		<form action="attributedemo9.php" method="get">
			Enter Your First Name: <input type="text" name="fn" /><br/>
			Enter Your Last Name: <input type="text" name="ln" /> <br/>
			<input type="submit" value="Submit to new tab" formtarget="_blank"/>
		</form>
	</body>
</html>
```

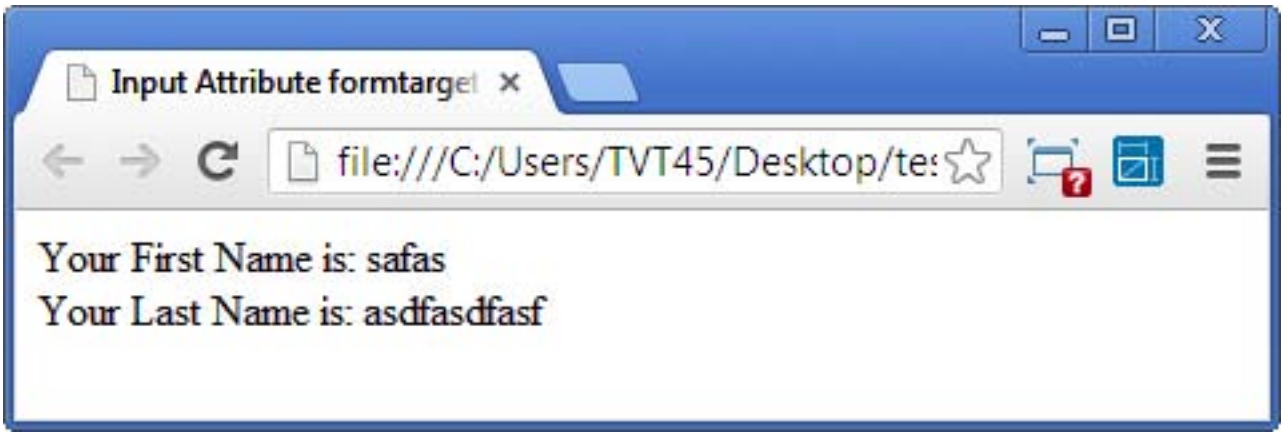
attributedemo9.php:

```
<html>
	<body>
		<?php
			$fn= $_GET["fn"];
			$ln= $_GET["ln"];
			echo "Your First Name is: " . $fn . "<br>";
			echo "Your Last Name is: " . $ln;
		?>
	</body>
</html>
```

Output:



On Clicking the Submit to new tab Button, the output will be displayed on a new tab:



- height and width: This attribute is used to set the height and width of the input element
  - Can only be used with input type: image
  - Syntax: <input type="image" height="" width="" />

Example for Attribute height and width:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Attribute height & width </title>
        </head>
        <body>
                <form action="attributedemo10.php" method="post">
                        <input type="image" src="index.jpg" height="150" width="150" />
                </form>
        </body>
</html>
```

Output:

- **List:** This attribute is used to refer to the <datalist> element. It specifies the id of the <datalist> element.
  - **Syntax:** <input list="" />

For Example refer to the datalist element.

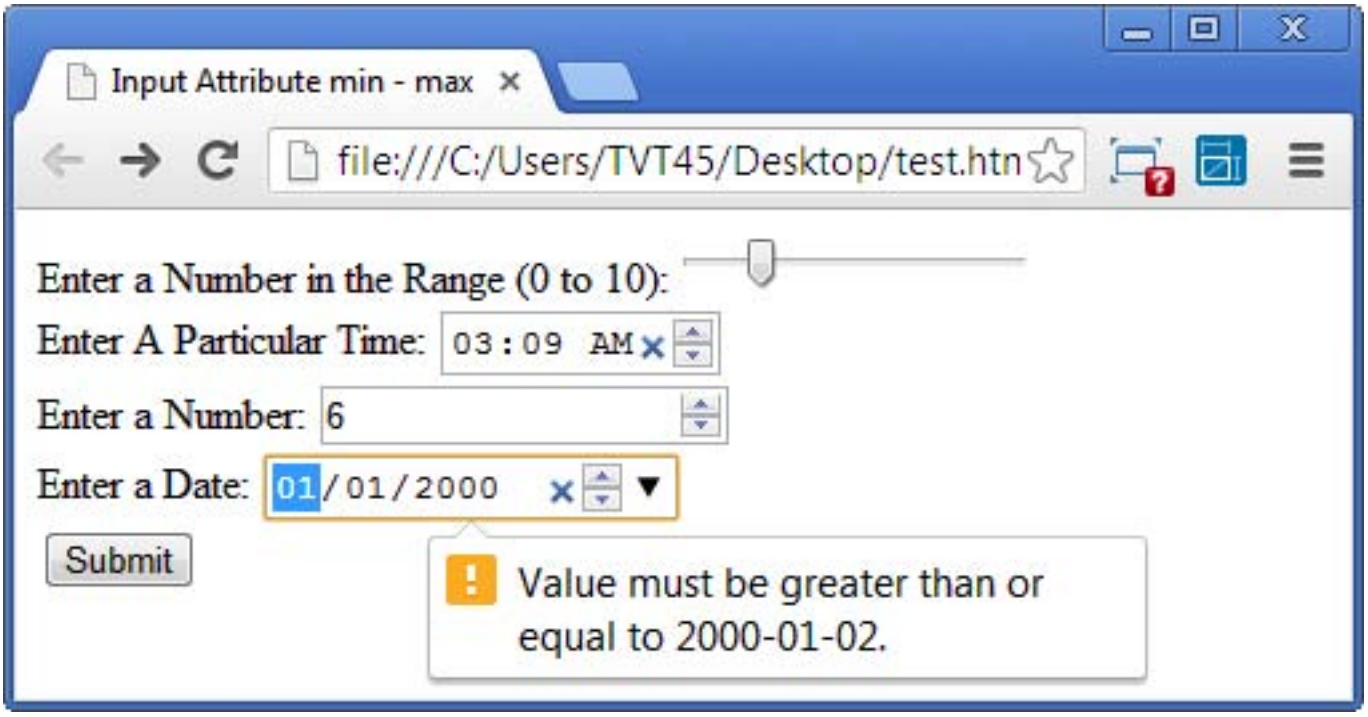
- min and max: These attributes are used to set the minimum and maximum value.
  - Can be used with the input types: number, range, date, datetime, datetime-local, month, time and week.
  - Syntax: <input type=" number/range/date/datetime/datetime-local/month/ time/week" min="" max="" />

Example for attribute min and max:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Attribute min - max </title>
        </head>
        <body>
                <form action="attributedemo11.php" method="post">
                        Enter a Number in the Range (0 to 10): <input type="range" name="range" min="0"
max="10"/> <br>
                        Enter A Particular Time: <input type="time" name="time" min="11:00 AM"
max="10:00 PM"/> <br>
                        Enter a Number: <input type="number" value="5" name="number" min="6"/> <br>
                        Enter a Date: <input type="date" name="date" min="2000-01-02" /> <br>
                        <input type="submit" value="Submit" />
                </form>
        </body>
</html>
```

Output:

When I clicked on the up arrow in Enter a Number field it went only till 6 not less than that. And when I entered the date 2000-01-01 It displayed :

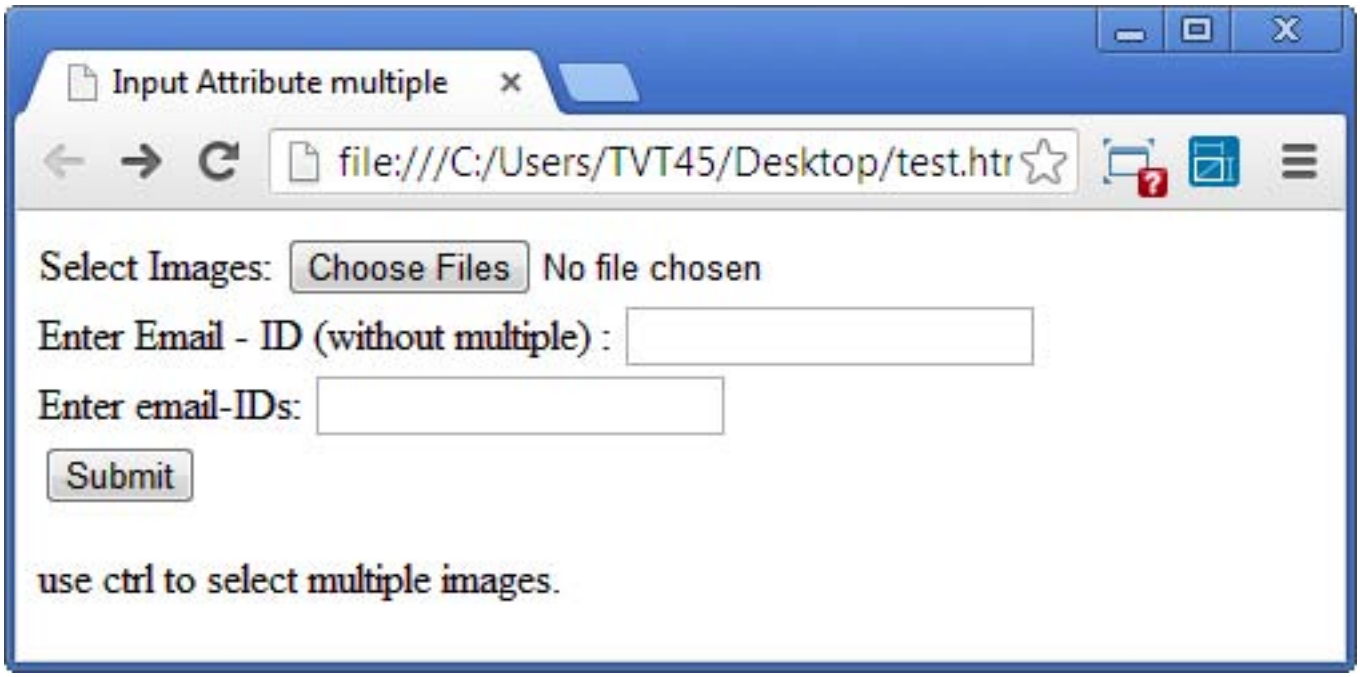


- **Multiple:** This attribute allows the user to select multiple values for a single input type.
  - Can only be used with input types: email, file.
  - Syntax: <input type="email/file" multiple />

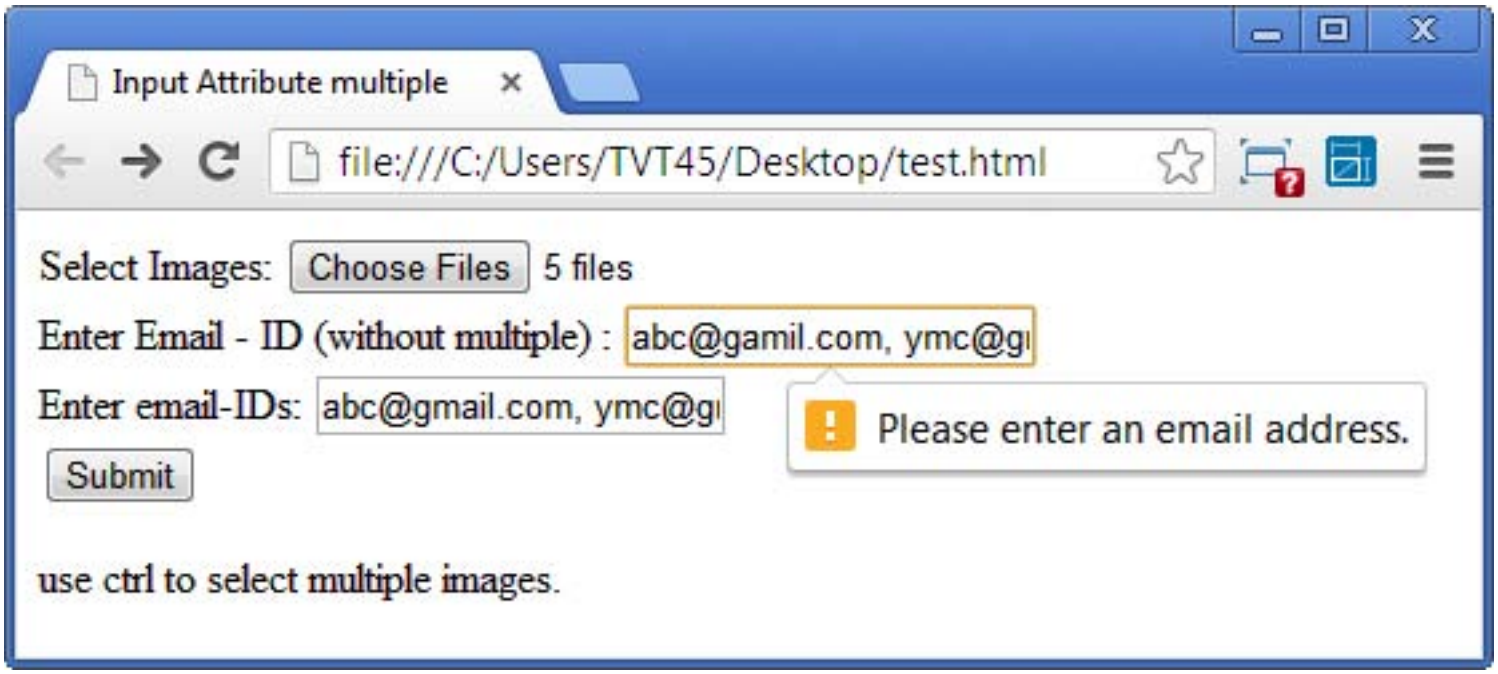
Example for attribute multiple:

```
<!DOCTYPE html>
<html>
	<head>
		<title> Input Attribute multiple </title>
	</head>
	<body>
		<form action="attributedemo11.php" method="post">
			Select Images: <input type="file" name="images" multiple /><br>
			Enter Email - ID (without multiple) : <input type="email" name="email" /><br>
			Enter email-IDs: <input type="email" name="email" multiple /><br>
			<input type="submit" value="Submit" />
		</form>
		<p> use ctrl to select multiple images. </p>
	</body>
</html>
```

Output:



Now on selecting 5 images, trying to enter 2 email ids in the textfield without multiple and entering 3 email-ids in the last textfield:

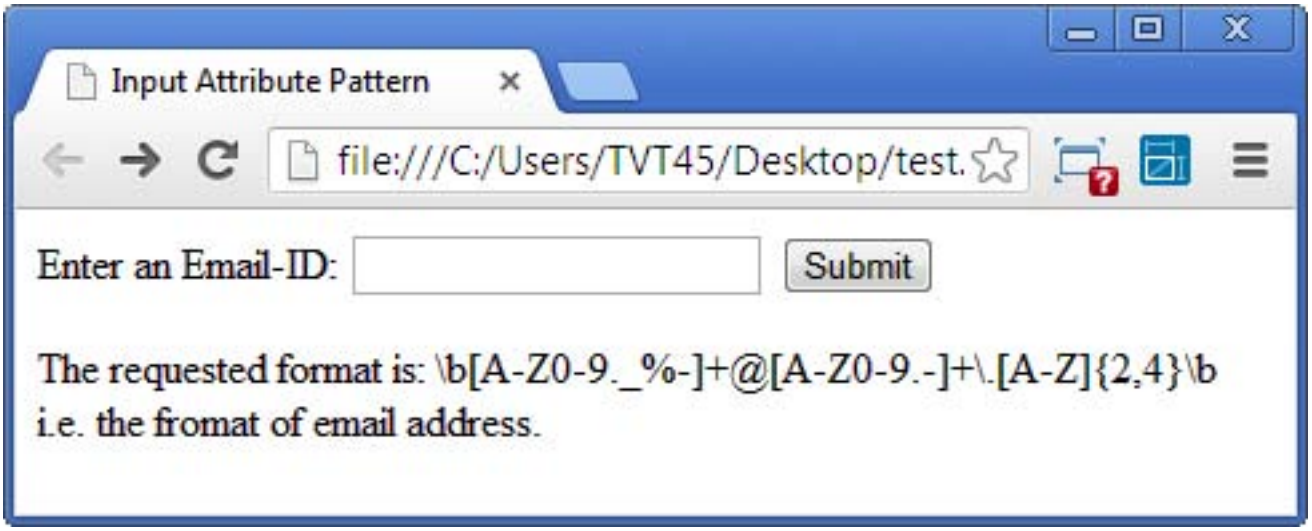


- pattern (regexp): This attribute specifies a regular expression that the <input> element's value is checked against.
  - Can only be used with input types: text, search, url, tel, email, and password.
  - **Syntax:** <input type=" text/search/url/tel/email/password" pattern="" />

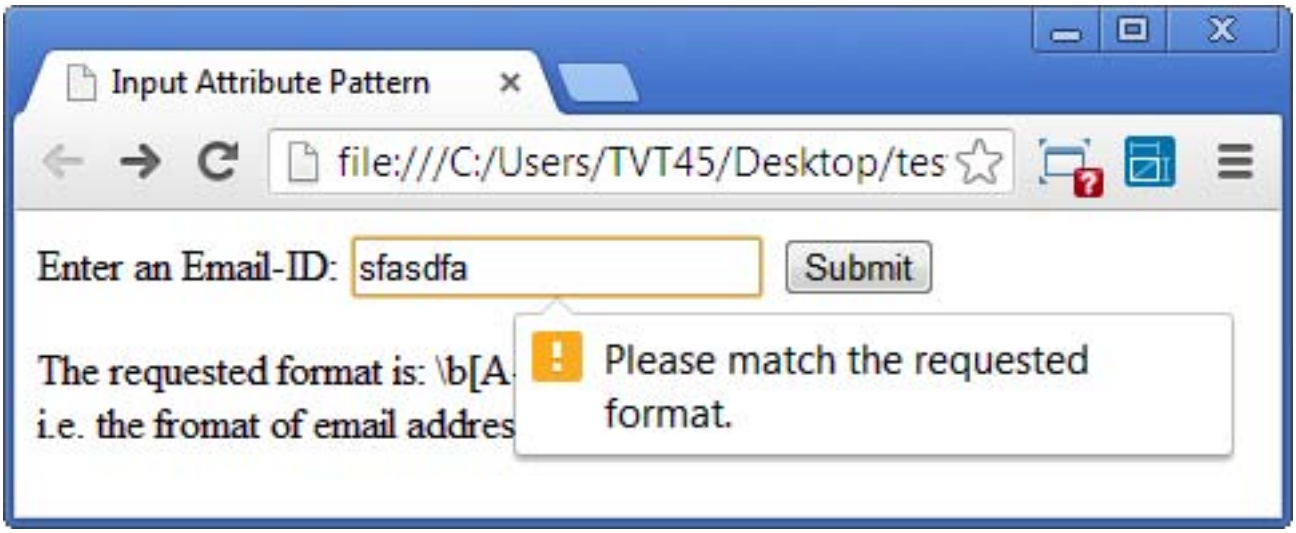
Example for attribute pattern:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Attribute Pattern </title>
        </head>
        <body>
                <form action="attributedemo2.php" method="post">
                        Enter an Email-ID: <input type="text" name="email"
pattern="\b[A-Z0-9._%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b" />
                        <input type="submit" value="Submit" />
                </form>
                <p> The requested format is: \b[A-Z0-9._%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b i.e. the fromat of email
address. </p>
        </body>
</html>
```

Output:



On entering an invalid value the output is:

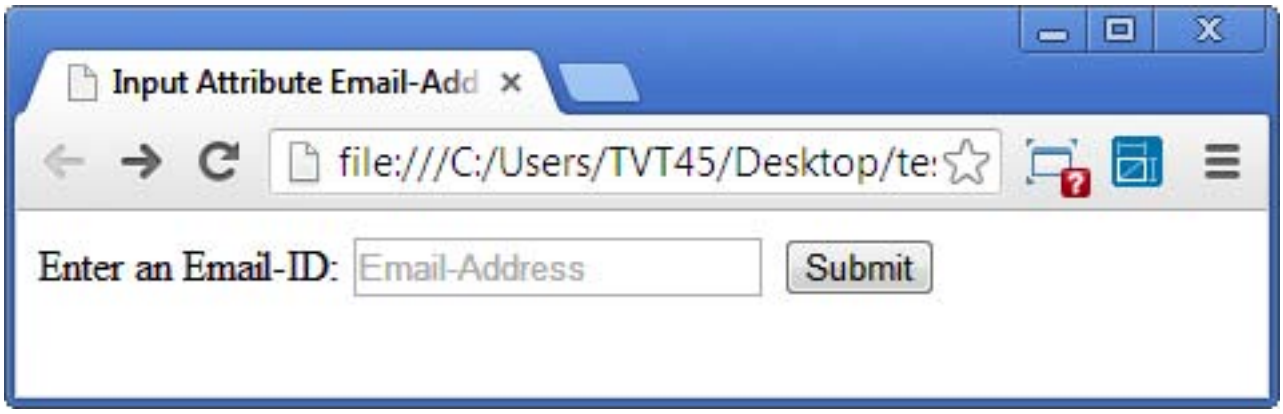


- Placeholder: The placeholder attribute specifies a short hint that describes the expected value of an input field.
- Can be used with input types: text, search, url, tel, email, and password.
- **Syntax:** <input type=" text/search/url/tel/email/password" placeholder="" />

Example of attribute placeholder:

```
<!DOCTYPE html>
<html>
	<head>
		<title> Input Attribute Email-Address </title>
	</head>
	<body>
		<form action="attributedemo2.php" method="post">
			Enter an Email-ID: <input type="email" name="email" placeholder="Email-Address"
/>
			<input type="submit" value="Submit" />
		</form>
	</body>
</html>
```

Output:



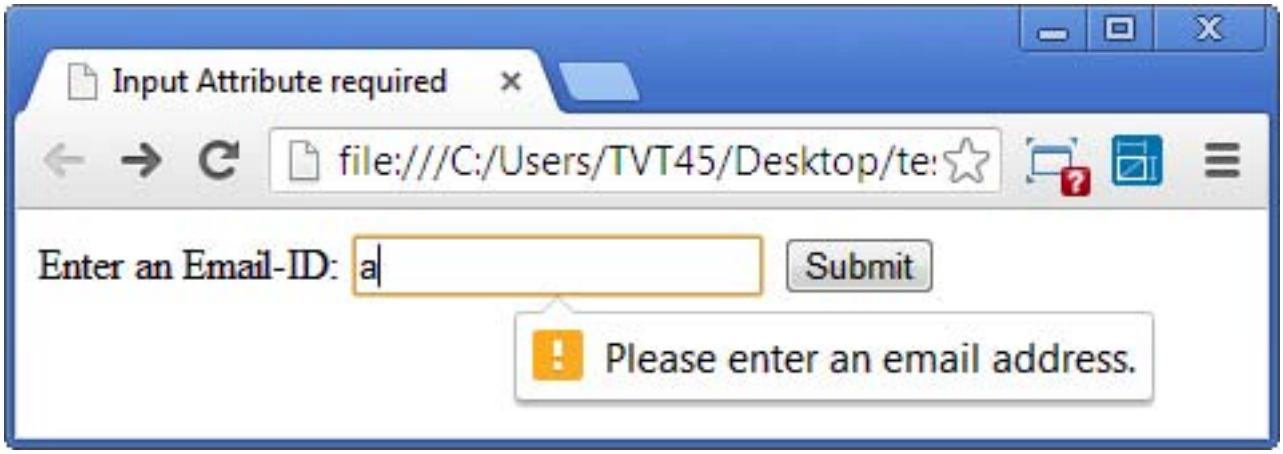
- **Required:** This attribute is used to specify the fields that are compulsorily needed to filled by the user.
  - Can be used with the input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.
  - Syntax: <input type=" text/search/url/tel/email/password/date pickers/number/checkbox/radio/file" required />

Example for attribute required:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Attribute required </title>
        </head>
        <body>
                <form action="attributedemo2.php" method="post">
                        Enter an Email-ID: <input type="email" name="email" required" />
                        <input type="submit" value="Submit" />
                </form>
        </body>
</html>
```

Output:

On Hitting the Submit Button without entering any value:

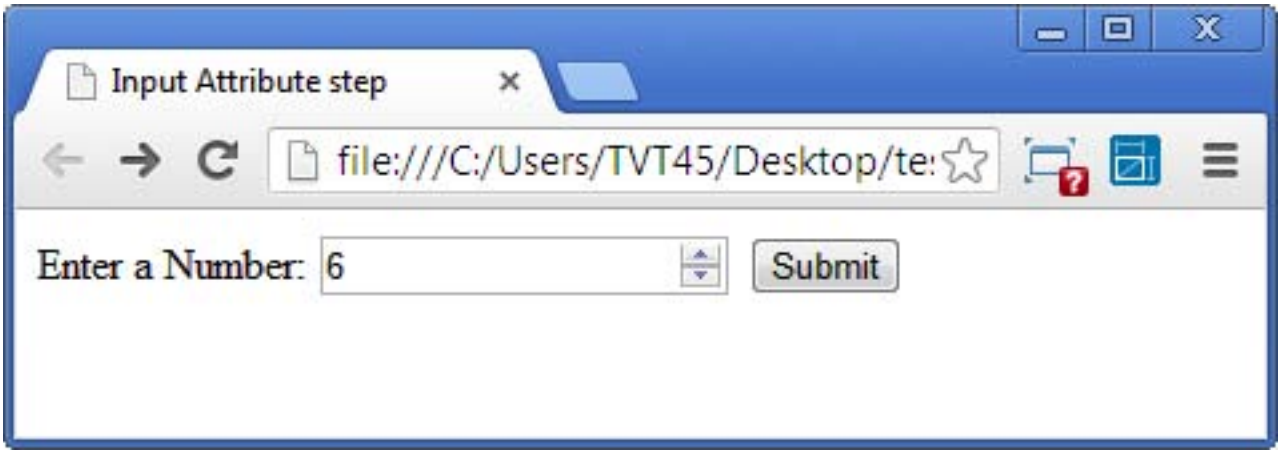


- Step: The step attribute specifies the legal number intervals for an <input> element.
  - Can be used with the input types: number, range, date, datetime, datetime-local, month, time and week.
  - Syntax: <input type=" number/range/date/datetime/datetime-local/month/time/week" step="" />

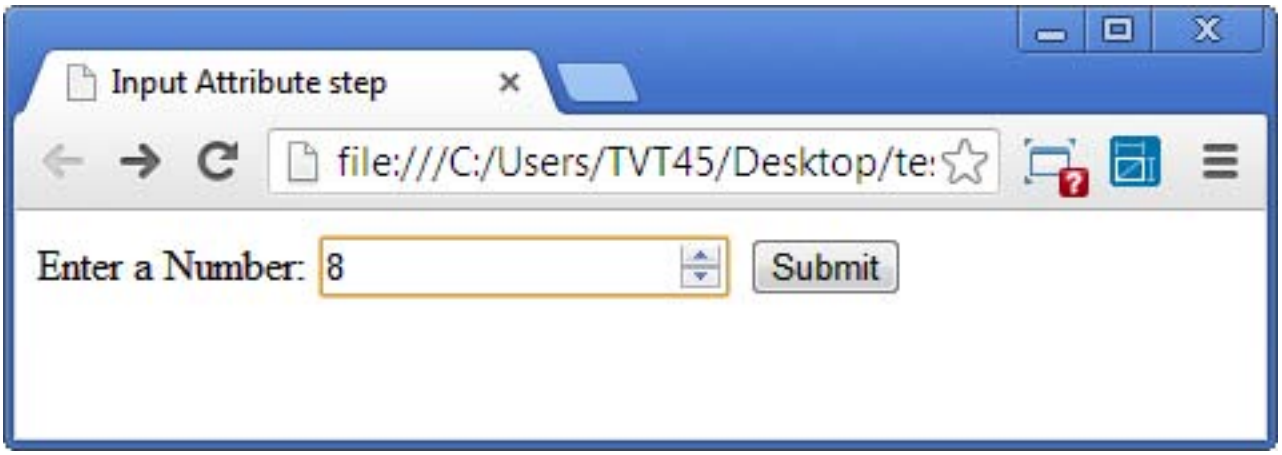
Example for attribute step:

```
<!DOCTYPE html>
<html>
        <head>
                <title> Input Attribute step </title>
        </head>
        <body>
                <form action="attributedemo.php" method="post">
                        Enter a Number: <input type="number" value="6" step="2" name="number"/>
                        <input type="submit" value="Submit" />
                </form>
        </body>
</html>
```

Output:



On clicking on the up arrow you will get a value 8 because we have set step to 2:



Thank You

True Vision
Technologies

Download Design Tips