HTML

By
Sudha Agarwal

INDEX

- Introduction
- HTML5 vs. HTML4
- Browser Support
- HTML5 Doctype
- Character Encoding
- New HTML5 Elements
- Semantic Elements
- New Form Elements
- New Input Types
- New Input Attributes
- HTML5 Canvas
- HTML5 SVG
- HTML5 Google Maps
- HTML5 Multimedia
- HTML5 Video
- HTML5 Audio
- HTML5 Geolocation
- HTML5 Drag and Drop
- HTML5 Local Storage

WHAT IS HTML5

- HTML5 is the new standard for HTML.

- The previous version of HTML was HTML 4.01, came in 1999.

- HTML5 is designed to deliver almost everything you want to do online without requiring additional plug-ins. It does everything from animation to apps, music to movies and can also be used to build complicated applications that run in your browser.

- HTML5 is also cross platform (it does not care whether you are using a tablet or a Smartphone, or a notebook or a smart TV).

Differences between HTML4 and HTML5

- HTML5 is a work in progress
- Simplified Syntax
- The new <canvas> Element for 2D drawing
- New Content specific elements like <article>, <header>, <footer>, <nav>, <section>
- No more <frame>, , <big>, <b>, <font>
- Support for local storage
- New <menu> and <figure> elements
- New <audio> and <video> Elements
- New form controls like calendar, date, time, email, url search.

HTML5 BROWSER SUPPORT

❖ HTML5 is not yet an official standard and no browser have full HTML5 support.

❖But all major browsers (Safari, Chrome, Firefox, Opera, Internet Explorer) continue to add new HTML5 features to their latest versions.

❖The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all have excellent support for HTML5.

HTML DOCTYPE

In HTML5 there is only one <!DOCTYPE> declaration and it is very simple.

```
<!DOCTYPE html>
```

CHARACTER ENCODING

- The character encoding (charset) declaration is also very simple:

```
<meta charset="UTF-8">
```

The default character encoding in HTML5 is UTF-8.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
Content of the document......
</body>

</html>
```

NEW HTML5 ELEMENTS

The most interesting new elements are:-

- New semantic elements like <header>, <footer>, <article>, and <section>.
- New form control attributes like number, date, time, calendar, and range.
- New graphic elements: <svg> and <canvas>.
- New multimedia elements: <audio> and <video>.

NEW HTML5 API'S

The most interesting new API's are:-

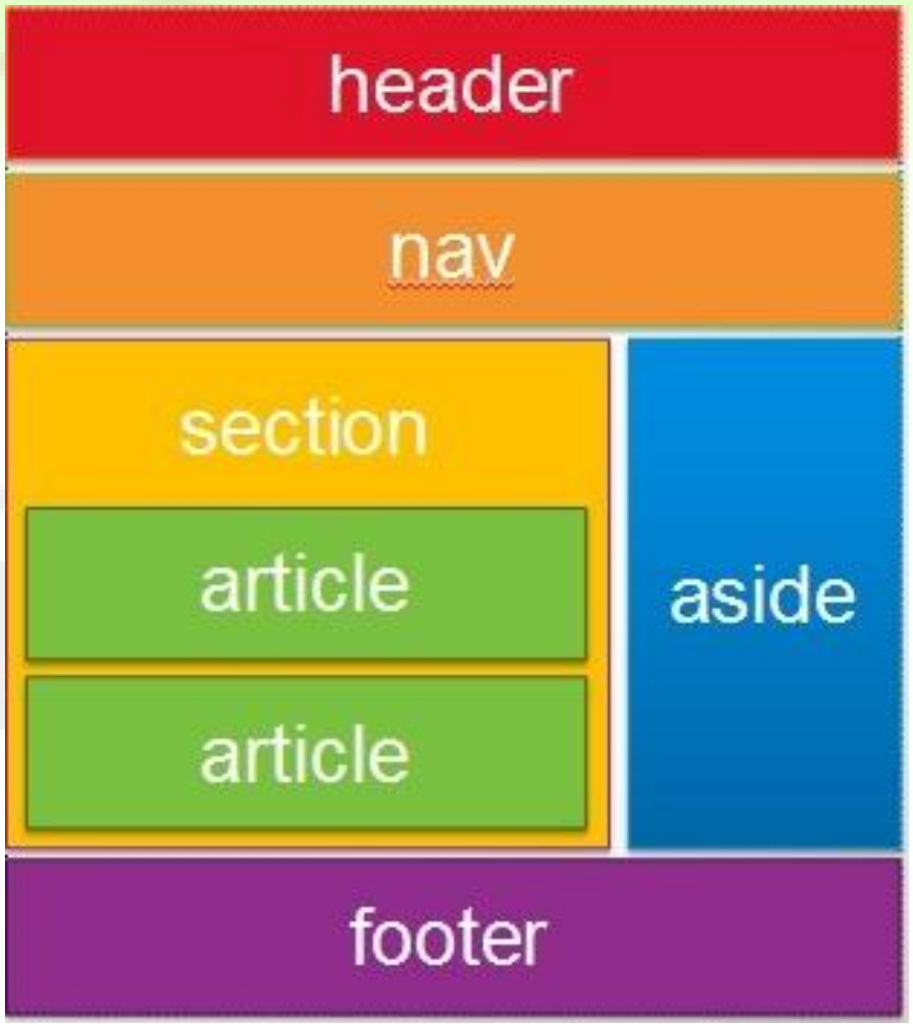
- HTML Geolocation
- HTML Drag and Drop
- HTML Local Storage
- HTML Application Cache
- HTML Web Workers
- HTML SSE

What are Semantic Elements?

- A semantic element clearly describes its meaning to both the browser and the developer.
- For example, and are non-semantic elements. They tell us nothing about their contents.
- But <form>, <table>, and <article> are semantic elements: They clearly define their content.
- HTML5 semantic elements are supported in all modern browsers.

NEW SEMANTIC/STRUCTURAL ELEMENTS

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



<SECTION> ELEMENT

- The <section> element defines a section in a document.
- A section is a thematic grouping of content, typically with a heading."

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF)
      is....</p>
</section>
```

<ARTICLE> ELEMENT

- The <article> element specifies independent, self-contained content.
- An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.
- Examples of where an <article> element can be used:
  - Forum post Blog post Newspaper article

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of
our planet's natural environment,
  and build a future in which humans live in
harmony with nature.</p>
</article>
```

<HEADER> ELEMENT

- The <header> element specifies a header for a document or section.

- The <header> element should be used as a container for introductory content.

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of
our planet's natural environment,
  and build a future in which humans live in
harmony with nature.</p>
</article>
```

<FOOTER> ELEMENT

- The <footer> element specifies a footer for a document or section.
- A <footer> element should contain information about its containing element.

```
<footer>
  <p>Posted by: someone</p>
  <p>Contact information: <a
href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

<NAV> ELEMENT

- The <nav> element defines a set of navigation links.

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

<aside> Element

- The <aside> element defines some content aside from the content it is placed in (like a sidebar).

- The aside content should be related to the surrounding content.

```
<p>My family and I visited The Epcot center this summer.</p>

<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

<figure> and <figcaption>

- The purpose of a figure caption is to add a visual explanation to an image.

- In HTML5, an image and a caption can be grouped together in a <figure> element:

```
<figure>
  <img src="pic_mountain.jpg" alt="The Pulpit
Rock" width="304" height="228">
  <figcaption>Fig1. - The Pulpit Rock,
Norway.</figcaption>
</figure>
```

<main> Element

- The <main> tag specifies the main content of a document.

- The <main> element must NOT be a descendant of an <article>, <aside>, <footer>, <header>, or <nav> element.

```
<main>
  <h1>Web Browsers</h1>
  <p>Google Chrome, Firefox, and Internet
Explorer are the most used browsers today.</p>
</main>
```

<details> and <summary>

- The <details> tag specifies additional details that the user can view or hide on demand.

- The <summary> tag defines a visible heading for the <details> element. The heading can be clicked to view/hide the details.

```
<details>
  <summary>Copyright 1999-2014.</summary>
  <p> - by Sudha. All Rights Reserved.</p>
  <p>All content and graphics on this
presentation are the property of Sudha.</p>
</details>
```

<details> and <summary>

- The <details> tag specifies additional details that the user can view or hide on demand.

- The <summary> tag defines a visible heading for the <details> element. The heading can be clicked to view/hide the details.

```
<details>
  <summary>Copyright 1999-2014.</summary>
  <p> - by Sudha. All Rights Reserved.</p>
  <p>All content and graphics on this
presentation are the property of Sudha.</p>
</details>
```

NEW FORM ELEMENTS

Tag	Description
<datalist>	Defines pre-defined options for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

<DATALIST> TAG

- The <datalist> tag specifies a list of pre-defined options for an <input> element.
- It is used to provide an "autocomplete" feature on <input> elements.
- Use the <input> element's list attribute to bind it together with a <datalist> element.

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

<KEYGEN> TAG

- The <keygen> tag specifies a key-pair generator field used for forms.

- When the form is submitted, the private key is stored locally, and the public key is sent to the server.

```
<form action="/action_page.php" method="get">
  Username: <input type="text" name="usr_name">
  Encryption: <keygen name="security">
  <input type="submit">
</form>
```

<OUTPUT> TAG

- The <output> tag represents the result of a calculation

```
<form
oninput="x.value=parseInt(a.value)+parseInt(b.va
lue)">0
  <input type="range" id="a" value="50">100
  +<input type="number" id="b" value="50">
  =<output name="x" for="a b"></output>
</form>
```

NEW INPUT TYPES

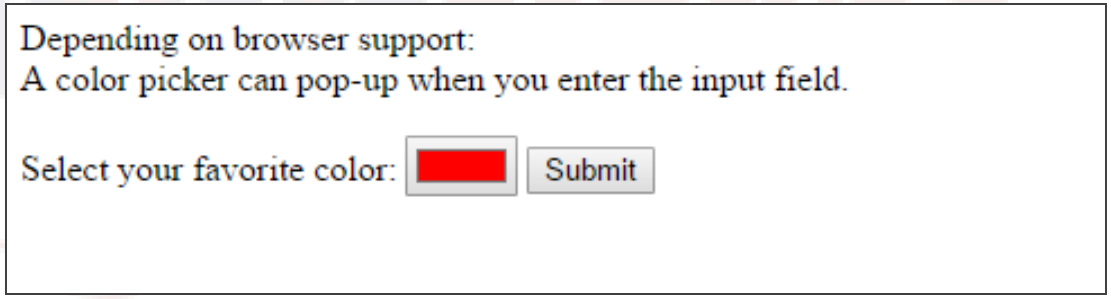
- color
- date
- datetime
- datetime-local
- email
- month
- week
- number
- range
- search
- tel
- time
- url

INPUT TYPE COLOR

- The <input type="color"> is used for input fields that should contain a color.

- Depending on browser support, a color picker can show up in the input field.

```
<form>
  Select your favorite color:
  <input type="color" name="favcolor">
</form>
```



INPUT TYPE DATE

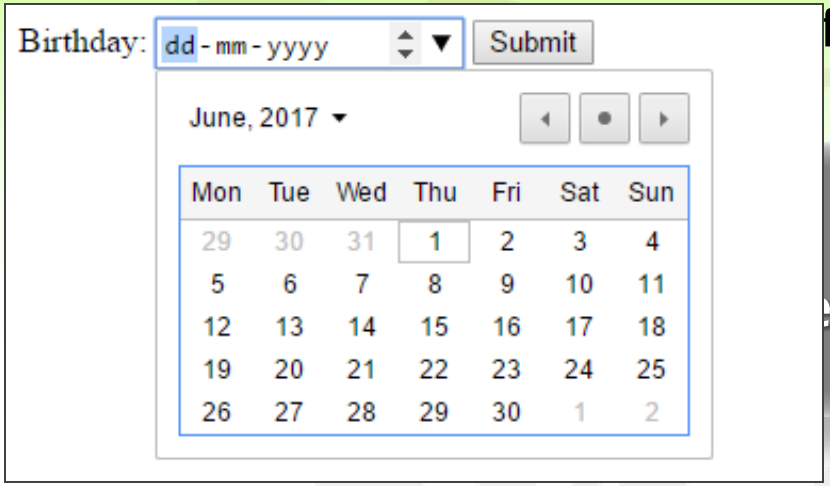
- The <input type="date"> is used for input fields that should contain a date.

```
<form>
  Birthday:
  <input type="date" name="bday">
</form>
```

```
<form>
  Enter a date before 1980-01-01:
  <input type="date" name="bday" max="1979-12-
31"><br>
  Enter a date after 2000-01-01:
  <input type="date" name="bday" min="2000-01-
02"><br>
</form>
```

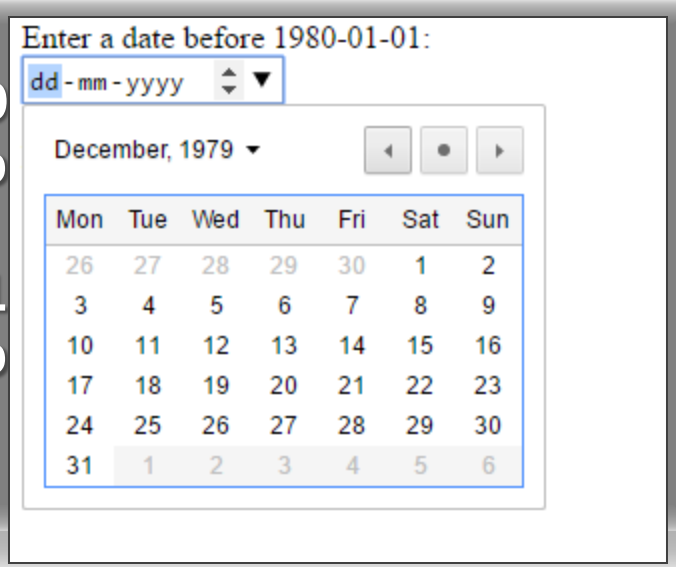
INPUT TYPE DATE

for input fields that should contain




```
e="bday">
```

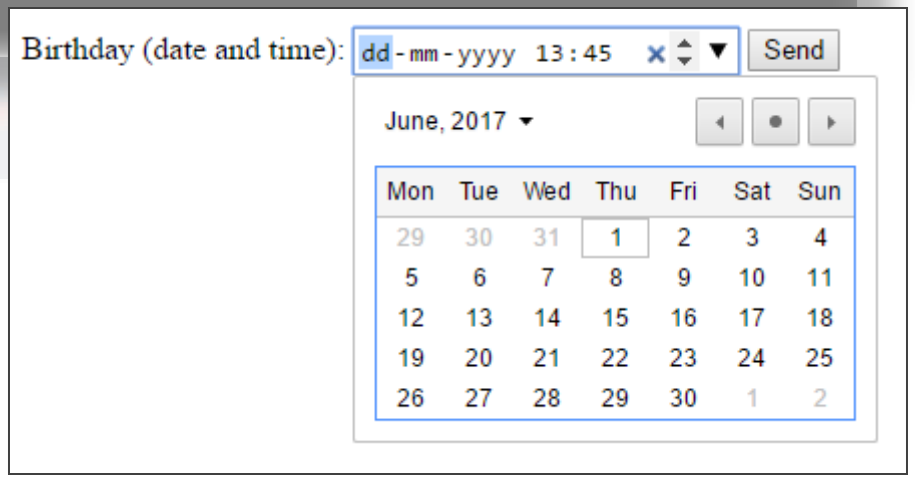
```
<form>
  Enter a date before 1980-0
  <input type="date" name="b
31"><br>
  Enter a date after 2000-01
  <input type="date" name="b
02"><br>
</form>
```



INPUT TYPE DATETIME-LOCAL

- The <input type="datetime-local"> specifies a date and time input field, with no time zone.

```
<form>
  Birthday (date and time):
  <input type="datetime-local" name="bdaytime">
</form>
```



INPUT TYPE EMAIL

- The <input type="email"> is used for input fields that should contain an e-mail address.

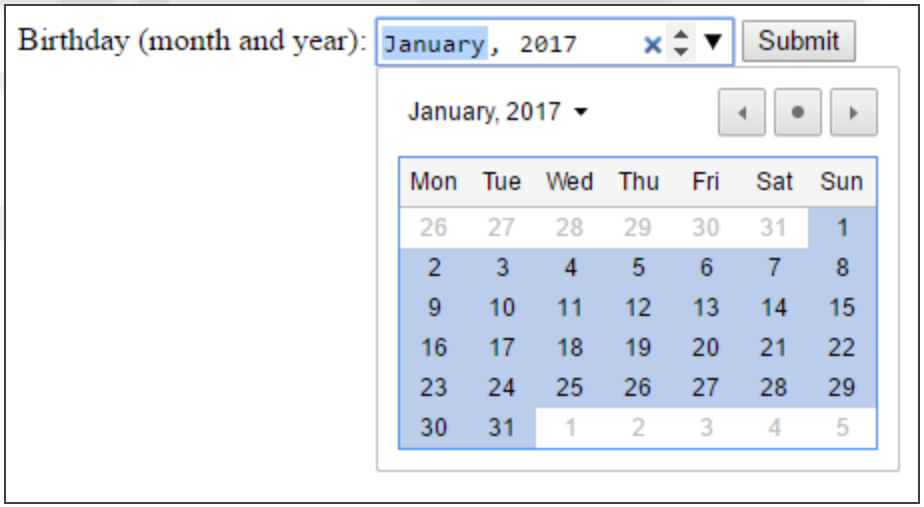
- Depending on browser support, the e-mail address can be automatically validated when submitted.

```
<form>
  E-mail:
  <input type="email" name="email">
</form>
```

INPUT TYPE MONTH

- The <input type="month"> allows the user to select a month and year.

```
<form>
  Birthday (month and year):
  <input type="month" name="bdaymonth">
</form>
```

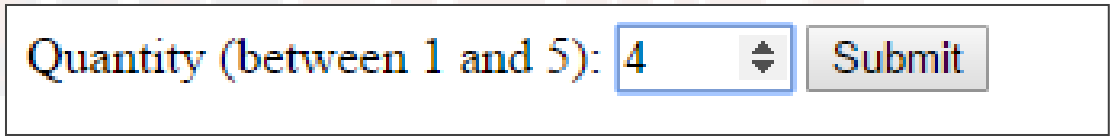


INPUT TYPE NUMBER

- The <input type="number"> defines a numeric input field.

- You can also set restrictions on what numbers are accepted.

```
<form>
  Quantity (between 1 and 5):
  <input type="number" name="quantity" min="1"
max="5">
</form>
```



INPUT RESTRICTIONS

Attribute	Restriction
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
Maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

INPUT RESTRICTIONS

```
<form>
  Quantity:
  <input type="number" name="points" min="0"
max="100" step="10" value="30">
</form>
```

INPUT TYPE RANGE

- The **<input type="range">** defines a control for entering a number whose exact value is not important (like a slider control).

- Default range is 0 to 100.

```
<form>
  <input type="range" name="points" min="0"
max="10">
</form>
```

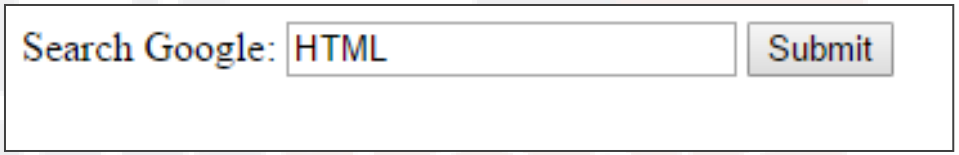
Attribute	Description
value	Defines the value of the slider
min	Minimum value of the range
Max	Maximum value of the range
step	This is Step scale factor of the slider, default value is 1

Example

INPUT TYPE SEARCH

- The **<input type="search">** is used for search fields (a search field behaves like a regular text field).

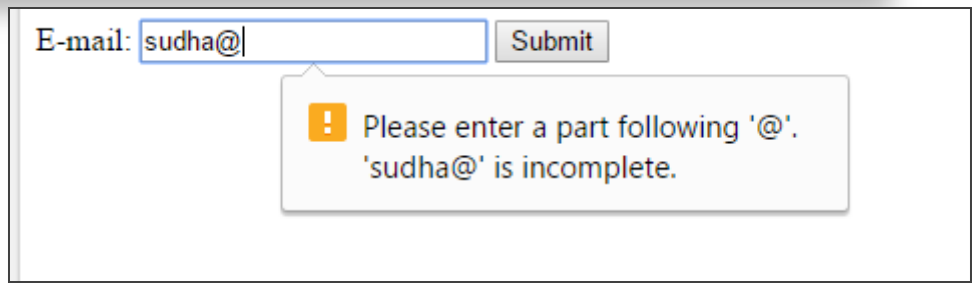
```
<form>
  Search Google:
  <input type="search" name="googlesearch">
</form>
```



INPUT TYPE EMAIL

- The **<input type="email">** is used for input fields that should contain an e-mail address.
- Depending on browser support, the e-mail address can be automatically validated when submitted.
- Some smartphones recognize the email type, and adds ".com" to the keyboard to match email input.

```
<form>
  E-mail:
  <input type="email" name="email">
</form>
```



INPUT TYPE TEL

- The <input type="tel"> is used for input fields that should contain a telephone number.

- The tel type is currently supported only in Safari 8.

```
<form>
  Telephone:
  <input type="tel" name="usrtel">
</form>
```

INPUT TYPE URL

- The <input type="url"> is used for input fields that should contain a URL address.

- Depending on browser support, the url field can be automatically validated when submitted.

```
<form>
  Add your homepage:
  <input type="url" name="homepage">
</form>
```

INPUT TYPE WEEK

- The <input type="week"> allows the user to select a week and year.

```
<form>
  Select a week:
  <input type="week" name="week_year">
</form>
```

NEW INPUT ATTRIBUTES

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

AUTOCOMPLETE ATTRIBUTE

- The autocomplete attribute specifies whether a form or input field should have autocomplete on or off.
- When autocomplete is on, the browser automatically complete values based on values that the user has entered before.
- The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.
- It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.

AUTOCOMPLETE ATTRIBUTE

```
<form action="/action_page.php" autocomplete="on">
  First name:<input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  E-mail: <input type="email" name="email"
autocomplete="off"><br>
  <input type="submit">
</form>
```

THE NOVALIDATE ATTRIBUTE

- The novalidate attribute is a <form> attribute.
- When present, novalidate specifies that the form data should not be validated when submitted.

```
<form action="/action_page.php" novalidate>
  E-mail: <input type="email" name="user_email">
  <input type="submit">
</form>
```

THE AUTOFOCUS ATTRIBUTE

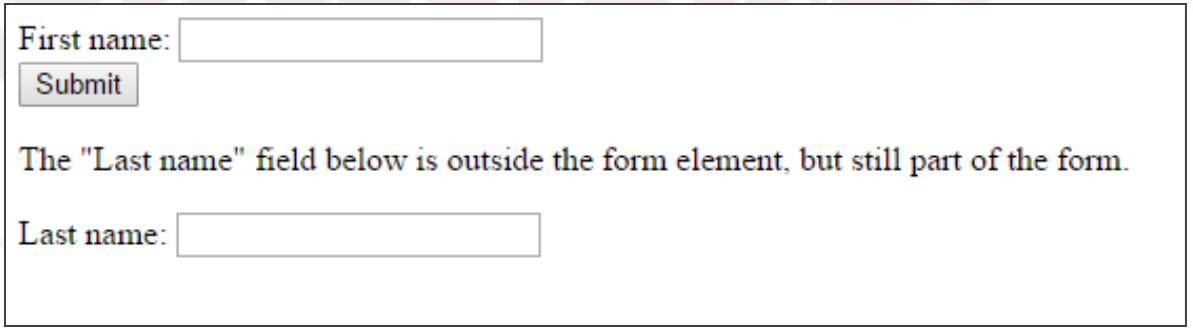
- The autofocus attribute specifies that the input field should automatically get focus when the page loads.

```
First name:<input type="text" name="fname" autofocus>
```

THE FORM ATTRIBUTE

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

```
<form action="/action_page.php" id="form1">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
</form>

Last name: <input type="text" name="lname"
form="form1">
```



THE FORMACTION ATTRIBUTE

- The formaction attribute specifies the URL of a file that will process the input control when the form is submitted.
- The formaction attribute overrides the action attribute of the <form> element.
- The formaction attribute is used with type="submit" and type="image".

```
<form action="/action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formaction="/action_page2.php"
  value="Submit as admin">
</form>
```

THE FORMMETHOD ATTRIBUTE

- The formmethod attribute defines the HTTP method for sending form-data to the action URL.
- The formmethod attribute overrides the method attribute of the <form> element.
- The formmethod attribute can be used with type="submit" and type="image".

```
<form action="/action_page.php" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formmethod="post"
formaction="action_page_post.asp"
  value="Submit using POST">
</form>
```

THE FORMTARGET ATTRIBUTE

- The formtarget attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.
- The formtarget attribute overrides the target attribute of the <form> element.
- The formtarget attribute can be used with type="submit" and type="image".

```
<form action="/action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit as normal">
  <input type="submit" formtarget="_blank"
  value="Submit to a new window">
</form>
```

FORMNOVALIDATE

- The formnovalidate attribute overrides the novalidate attribute of the <form> element.
- The formnovalidate attribute can be used with type="submit".

```
<form action="/action_page.php">
  E-mail: <input type="email" name="userid"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formnovalidate value="Submit
without validation">
</form>
```

HEIGHT AND WIDTH

- The height and width attributes specify the height and width of an <input type="image"> element.

```
<input type="image" src="img_submit.gif" alt="Submit"
width="48" height="48">
```

THE LIST ATTRIBUTE

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

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

MIN AND MAX

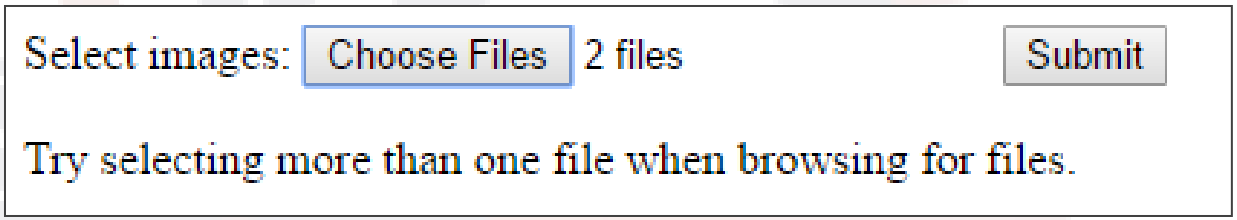
- The min and max attributes specify the minimum and maximum values for an <input> element.
- The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.

```
Enter a date before 1980-01-01:
<input type="date" name="bday" max="1979-12-31">

Enter a date after 2000-01-01:
<input type="date" name="bday" min="2000-01-02">

Quantity (between 1 and 5):
<input type="number" name="quantity" min="1" max="5">
```

THE MULTIPLE ATTRIBUTE

- The multiple attribute specifies that the user is allowed to enter more than one value in the <input> element.
- The multiple attribute works with the following input types: email, and file.

```
Select images: <input type="file" name="img" multiple>
```



THE PATTERN ATTRIBUTE

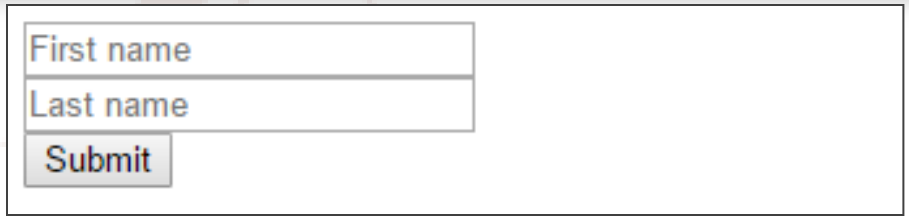
- The pattern attribute specifies a regular expression that the <input> element's value is checked against.
- The pattern attribute works with the following input types: text, search, url, tel, email, and password.

```
Country code: <input type="text" name="country_code"
pattern="[A-Za-z]{3}" title="Three letter country
code">
```

THE PLACEHOLDER ATTRIBUTE

- The placeholder attribute specifies a hint that describes the expected value of an input field (a sample value or a short description of the format).
- The hint is displayed in the input field before the user enters a value.
- The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

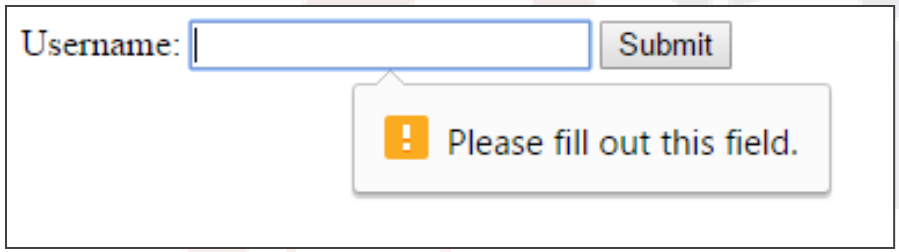
```
<input type="text" name="fname" placeholder="First name">
```



THE REQUIRED ATTRIBUTE

- The required attribute specifies that an input field must be filled out before submitting the form.
- The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

```
Username: <input type="text" name="usrname" required>
```



THE STEP ATTRIBUTE

- The step attribute specifies the legal number intervals for an <input> element.
- Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.
- The step attribute works with the following input types: number, range, date, datetime-local, month, time and week.

```
<input type="number" name="points" step="3">
```

HTML5 CANVAS

HTML5 CANVAS

- The HTML **<canvas>** element is used to draw graphics, on the fly, via JavaScript.
- The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Element					
<canvas>	4.0	9.0	2.0	3.1	9.0

CANVAS EXAMPLE

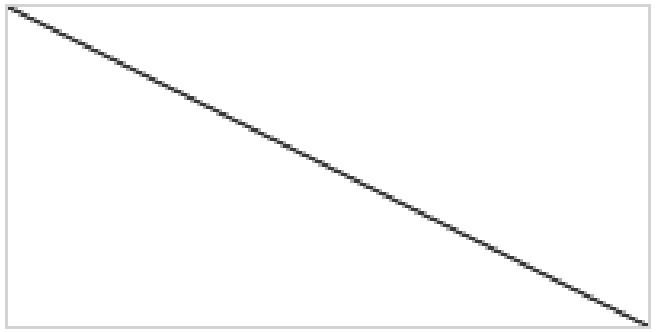
- A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.
- The markup looks like this:

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

- Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas. To add a border, use the style attribute.

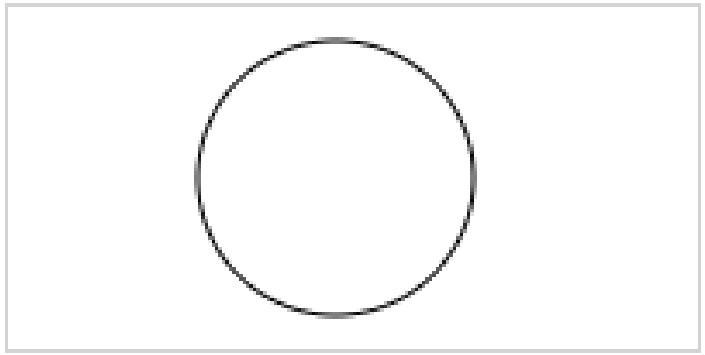
DRAW A LINE

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
```



DRAW A CIRCLE

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
```



DRAW A TEXT

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
```

CANVAS GRADIENTS

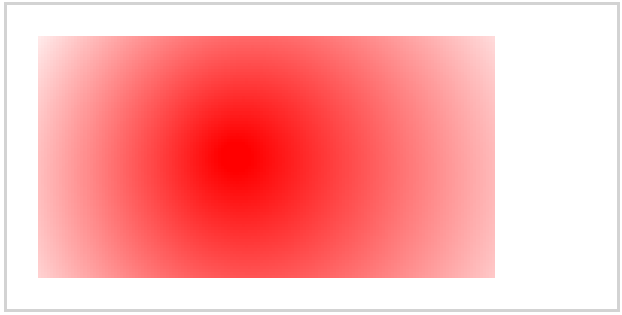
- Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors.
- There are two different types of gradients:
  - createLinearGradient(x,y,x1,y1) - creates a linear gradient
  - createRadialGradient(x,y,r,x1,y1,r1) - creates a radial/circular gradient
- Once we have a gradient object, we must add two or more color stops.
- The addColorStop() method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.
- To use the gradient, set the fillStyle or strokeStyle property to the gradient, then draw the shape (rectangle, text, or a line).

LINEAR GRADIENT

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");

// Create gradient
var grd=ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle=grd;
ctx.fillRect(10,10,150,80);
```

RADIAL GRADIENT

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");

// Create gradient
var grd=ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
```



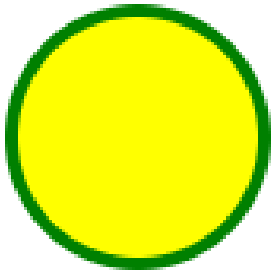
HTML5 SVG

HTML5 SVG

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation
- The HTML <svg> element is a container for SVG graphics.
- SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

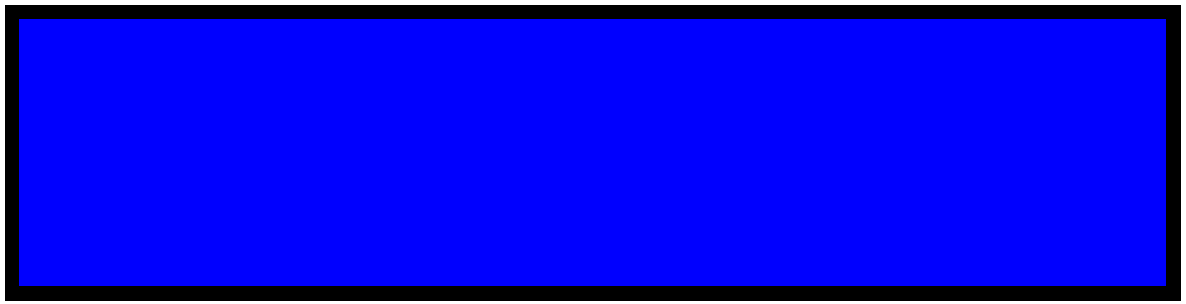
SVG CIRCLE

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

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green"
stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```



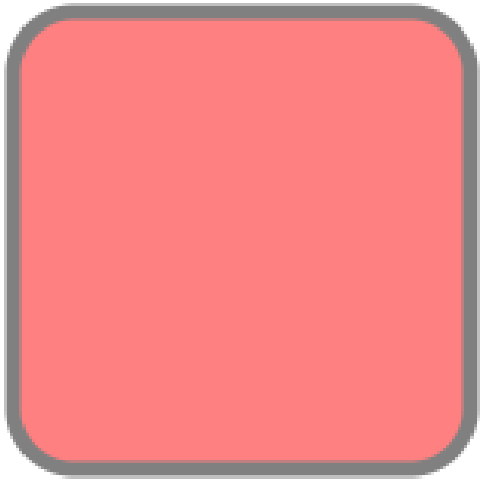
SVG RECTANGLE

```
<svg width="400" height="100">
  <rect width="400" height="100"
style="fill:rgb(0,0,255); stroke-width:10;
stroke:rgb(0,0,0)" />
</svg>
```



SVG ROUNDED RECTANGLE

```
<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150"
height="150"
  style="fill:red;stroke:black;stroke-
width:5;opacity:0.5" />
</svg>
```



Differences between SVG and Canvas

- SVG is a language for describing 2D graphics in XML.
- Canvas draws 2D graphics, on the fly (with a JavaScript).
- SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.
- In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.
- Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

Comparison of Canvas and SVG

❖Canvas

❖Resolution dependent

❖No support for event handlers

❖Poor text rendering capabilities

❖You can save the resulting image as .png or .jpg

❖Well suited for graphic-intensive games

❖SVG

❖Resolution independent

❖Support for event handlers

❖Best suited for applications with large rendering areas (Google Maps)

❖Slow rendering if complex (anything that uses the DOM a lot will be slow)

❖Not suited for game applications

GOOGLE MAP

HTML GOOGLE MAPS

- Google Maps allows you to display maps on your web page:

HTML GOOGLE MAPS

- To demonstrate how to add a Google Map to a web page, we will use a basic HTML page

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

<h1>My First Google Map</h1>

<div id="map">My map will go here</div>

</body>
<html>
```

HTML GOOGLE MAPS

- Set the size of the map

```
<div id="map" style="width:400px;height:400px">
```

HTML GOOGLE MAPS

- Create a Function to Set The Map Properties

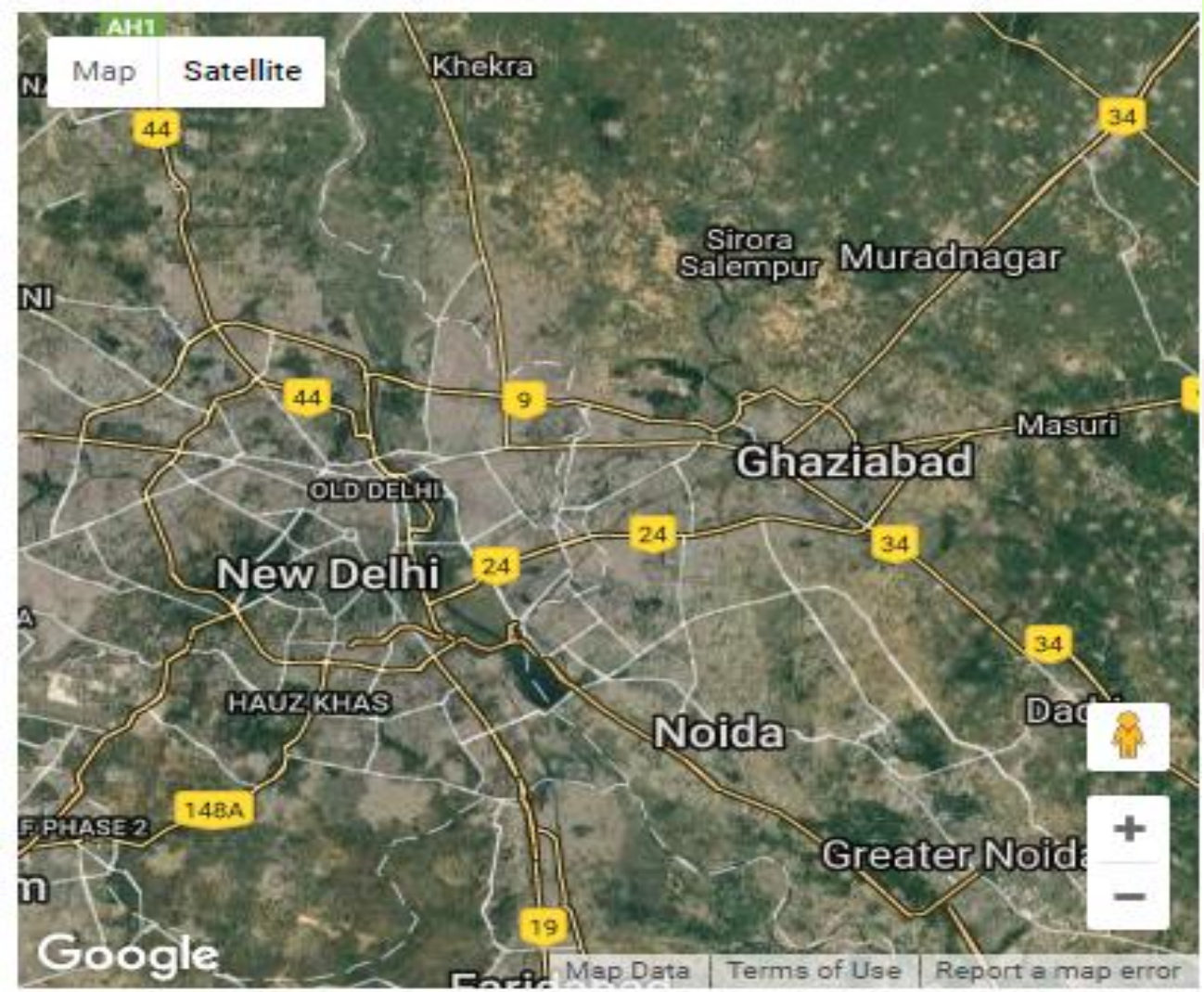
```
function myMap() {
    var mapOptions = {
        center: new google.maps.LatLng(28.65, 77.34),
        zoom: 10,
        mapTypeId: google.maps.MapTypeId.HYBRID
    }
var map = new
google.maps.Map(document.getElementById("map"),
mapOptions);
}
```

HTML GOOGLE MAPS

- Finally, show the map on the page!
- The functionality of the map is provided by a JavaScript library located at Google. Add a script to refer to the Google Maps API with a callback to the myMap function:

```
<script
src="https://maps.googleapis.com/maps/api/js?callback=
myMap"></script>
```

HTML GOOGLE MAPS



HTML5
MULTIMEDIA

HTML MULTIMEDIA

- Multimedia comes in many different formats. It can be almost anything you can hear or see.
- Examples: Images, music, sound, videos, records, films, animations, and more.
- Web pages often contain multimedia elements of different types and formats.

MULTIMEDIA FORMATS

- Multimedia elements (like audio or video) are stored in media files.
- The most common way to discover the type of a file, is to look at the file extension.
- Multimedia files have formats and different extensions like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

HTML5 Video

- Before HTML5, a video could only be played in a browser with a plug-in (like flash).
- The HTML5 <video> element specifies a standard way to embed a video in a web page.

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

HTML5 VIDEO

- To start a video automatically use the autoplay attribute

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

HTML Video – Browser Support

- In HTML5, there are 3 supported video formats: MP4, WebM, and Ogg.
- The browser support for the different formats is:

Browser	MP4	WebM	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	NO	NO
Opera	YES (from Opera 25)	YES	YES

HTML5 AUDIO

- The HTML5 <audio> element specifies a standard way to embed audio in a web page.

- To play an audio file in HTML, use the <audio> element

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

HTML5 Audio – How it Works

- The controls attribute adds audio controls, like play, pause, and volume.
- The **<source>** element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.
- The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

HTML AUDIO – BROWSER SUPPORT

- In HTML5, there are 3 supported audio formats: MP3, Wav, and Ogg.
- The browser support for the different formats is:

Browser	MP3	Wav	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

HTML5 GEOLOCATION

HTML5 API - GEOLOCATION

- The HTML Geolocation API is used to get the geographical position of a user.
- Since this can compromise privacy, the position is not available unless the user approves it.

USING HTML GEOLOCATION

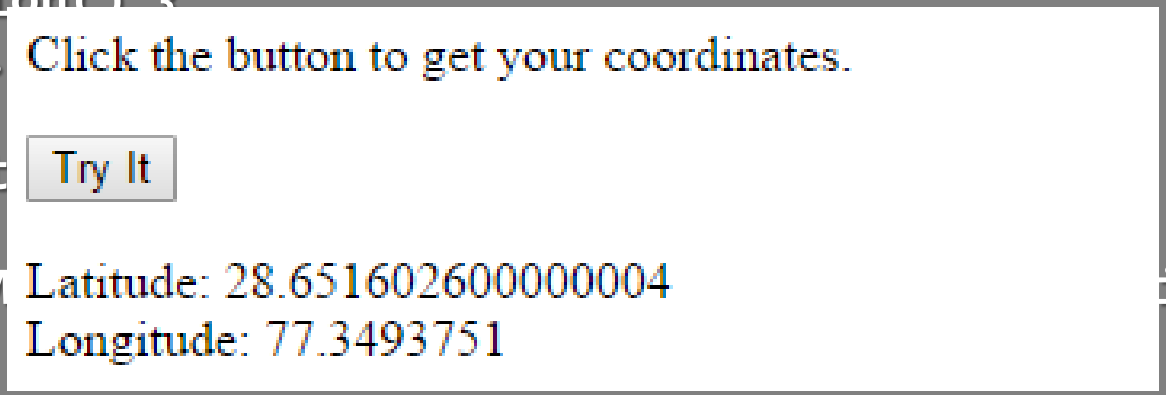
- The getCurrentPosition() method is used to return the user's position.

```
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(showPosition);
    } else {
        x.innerHTML = "Geolocation is not supported by this
browser.";
    }
}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

USING HTML GEOLOCATION

- The getCurrentPosition() method is used to return the user's position.

```
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.

navigator.geolocat
    } else {
        x.innerHTM                                          is
browser.";
    }
}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```



HANDLING ERRORS

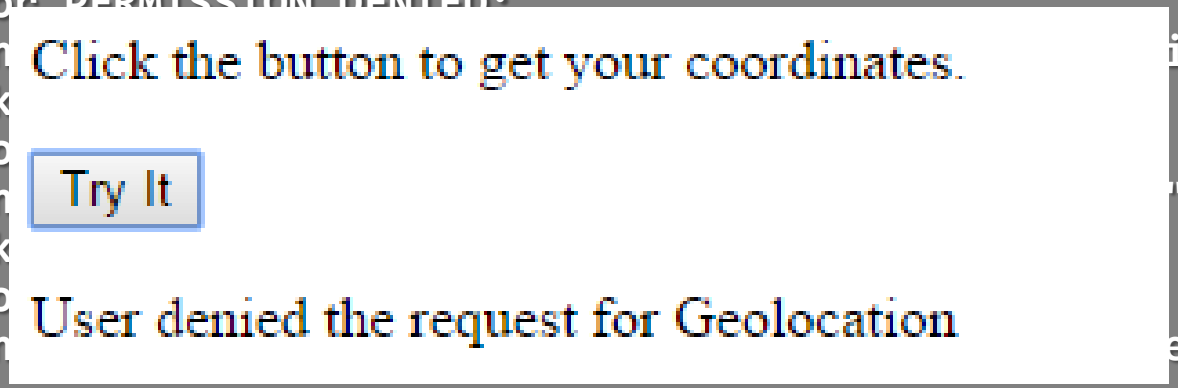
- The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location:

```
function showError(error) {
    switch(error.code) {
        case error.PERMISSION_DENIED:
            x.innerHTML = "User denied the request for Geolocation."
            break;
        case error.POSITION_UNAVAILABLE:
            x.innerHTML = "Location information is unavailable."
            break;
        case error.TIMEOUT:
            x.innerHTML = "The request to get user location timed
out."
            break;
        case error.UNKNOWN_ERROR:
            x.innerHTML = "An unknown error occurred."
            break;
    }
}
```

HANDLING ERRORS

- The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location:

```
function showError(error) {
    switch(error.code) {
        case error.PERMISSION_DENIED:
            x.inn                                                    ion."
            break
        case erro
            x.inn                                                    "
            break
        case erro
            x.inn                                                    ed
out."
            break;
        case error.UNKNOWN_ERROR:
            x.innerHTML = "An unknown error occurred."
            break;
    }
}
```



DISPLAY THE RESULT IN A MAP

- To display the result in a map, you need access to a map service, like Google Maps.
- In the example below, the returned latitude and longitude is used to show the location in a Google Map (using a static image):

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

    var img_url =
"https://maps.googleapis.com/maps/api/staticmap?center=

"+latlon+"&zoom=14&size=400x300&sensor=false&key=YOUR_:KEY";

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

DISPLAY THE RESULT IN A MAP

- To display the result in a map, you need access to a map service, like Google Maps.
- In the example be ed to show the locatic


```
function showPosit
    var latlon = p
position.coords.lo

    var img_url =
"https://maps.goog                                    r=

"+latlon+"&zoom=14                                    _:KEY";

    document.getEl                                    "<img
src='"+img_url+"'>
}
```

LOCATION-SPECIFIC INFORMATION

- This page has demonstrated how to show a user's position on a map.
- Geolocation is also very useful for location-specific information, like:
  - Up-to-date local information
  - Showing Points-of-interest near the user
  - Turn-by-turn navigation (GPS)

GETCURRENTPOSITION() METHOD

- The getCurrentPosition() method returns an object on success.
- The latitude, longitude and accuracy properties are always returned.

Property	Returns
coords.latitude	The latitude as a decimal number (always returned)
coords.longitude	The longitude as a decimal number (always returned)
coords.accuracy	The accuracy of position (always returned)
coords.altitude	The altitude in meters above the mean sea level (returned if available)
coords.altitudeAccuracy	The altitude accuracy of position (returned if available)
coords.heading	The heading as degrees clockwise from North (returned if available)
coords.speed	The speed in meters per second (returned if available)
timestamp	The date/time of the response (returned if available)

OTHER INTERESTING METHODS

- The Geolocation object also has other interesting methods:
  - watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
  - clearWatch() - Stops the watchPosition() method.

OTHER INTERESTING METHODS

```
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.watchPosition(showPosition);
    } else {
        x.innerHTML = "Geolocation is not supported by this
browser.";
    }
}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

OTHER INTERESTING METHODS

```
<script>
var x = document.getElementById("demo");
function getLocation()
    if (navigator.geolo
        navigator.geolo                                    on);
    } else {
        x.innerHTML = "                                    this
browser.";
    }
}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

Click the button to get your coordinates.

Try It

Latitude: 28.651578299999997
Longitude: 77.34938079999999

HTML5 DRAG AND DROP

HTML5 DRAG AND DROP

- Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.
- In HTML5, drag and drop is part of the standard: Any element can be draggable.

HTML5 DRAG AND DROP

```
<head>
<script>
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69">

</body>
```

HTML5 DRAG AND DROP

- First of all: To make an element draggable, set the draggable attribute to true:

```
<img draggable="true">
```

- Then, specify what should happen when the element is dragged.
- In the example above, the ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.
- The dataTransfer.setData() method sets the data type and the value of the dragged data:

```
function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}
```

HTML5 DRAG AND DROP

- The **ondragover** event specifies where the dragged data can be dropped.
- By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.
- This is done by calling the **event.preventDefault()** method for the ondragover event:

```
event.preventDefault()
```

HTML5 DRAG AND DROP

- When the dragged data is dropped, a drop event occurs.
- In the example above, the **ondrop** attribute calls a function, drop(event):

```
function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");

ev.target.appendChild(document.getElementById(data));
}
```

HTML5
LOCALSTORAGE

HTML5 LOCAL STORAGE

- With local storage, web applications can store data locally within the user's browser.
- Before HTML5, application data had to be stored in cookies, included in every server request. Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
- Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
- Local storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

HTML LOCAL STORAGE OBJECTS

- HTML local storage provides two objects for storing data on the client:
  - **window.localStorage** - stores data with no expiration date
  - **window.sessionStorage** - stores data for one session (data is lost when the browser tab is closed)
- Before using local storage, check browser support for localStorage and sessionStorage:

```
if (typeof(Storage) !== "undefined") {
    // Code for localStorage/sessionStorage.
} else {
    // Sorry! No Web Storage support..
}
```

THE LOCALSTORAGE OBJECT

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

```
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML =
localStorage.getItem("lastname");
```

THE LOCALSTORAGE OBJECT

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

```
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML =
localStorage.getItem("lastname");
```

- The syntax for removing the "lastname" localStorage item is as follows:

```
localStorage.removeItem("lastname");
```

THE LOCALSTORAGE OBJECT

- The following example counts the number of times a user has clicked a button

```
if (localStorage.clickcount) {
    localStorage.clickcount =
Number(localStorage.clickcount) + 1;
} else {
    localStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "You
have clicked the button " +
localStorage.clickcount + " time(s).";
```

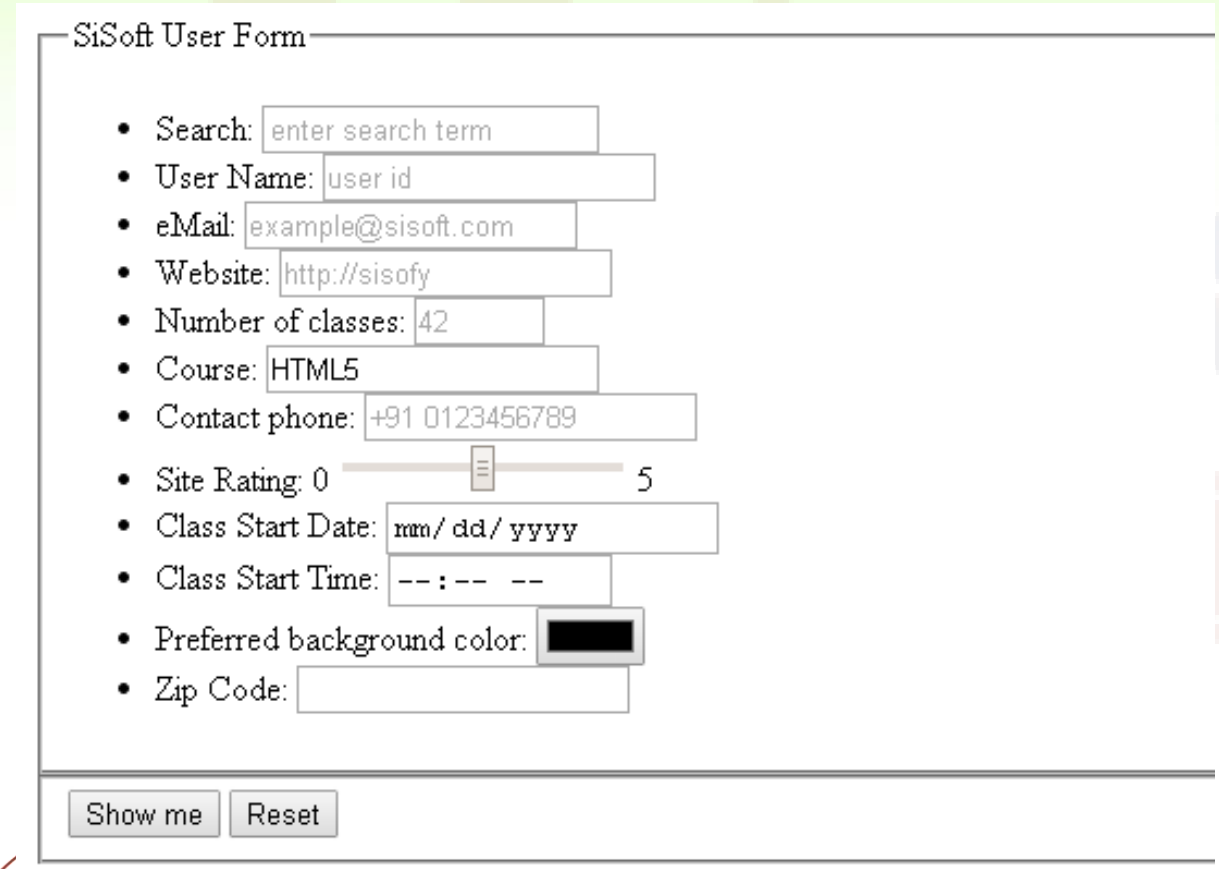
THE SESSIONSTORAGE OBJECT

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

```
if (sessionStorage.clickcount) {
    sessionStorage.clickcount =
Number(sessionStorage.clickcount) + 1;
} else {
    sessionStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "You
have clicked the button " +
sessionStorage.clickcount + " time(s) in this
session.";
Try it Yourself »
```

ASSIGNMENT

- Create a User detail form with validations as shown below:



Thank you!

Contact Us

info@4kitsolutions.com