

1. Introduction
2. Advanced Input types
3. SVG Tag
4. Canvas Tag
5. Audio Tag
6. Video Tag
7. Geolocation
8. **Introduction**

New HTML 5 Elements

* New elements HEADER Tag, FOOTER Tag, ARTICLE Tag, and SECTION tag.
* New form control attributes like number, date, time, calendar, and range.
* New graphic elements: SVG Tag and CANVAS Tag
* New multimedia elements: AUDIO Tag and VIDEO Tag

New Semantic/Structural Elements

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <article> | Defines an article in the document |
| <aside> | Defines content aside from the page content |
| <bdi> | Defines a part of text that might be formatted in a different direction from other text |
| <details> | Defines additional details that the user can view or hide |
| <dialog> | Defines a dialog box or window |
| <figcaption> | Defines a caption for a <figure> element |
| <figure> | Defines self-contained content, like illustrations, diagrams, photos, code listings, etc. |
| <footer> | Defines a footer for the document or a section |
| <header> | Defines a header for the document or a section |
| <main> | Defines the main content of a document |
| <mark> | Defines marked or highlighted text |
| <menuitem>  | Defines a command/menu item that the user can invoke from a popup menu |
| <meter> | Defines a scalar measurement within a known range (a gauge) |
| <nav> | Defines navigation links in the document |
| <progress> | Defines the progress of a task |
| <rp> | Defines what to show in browsers that do not support ruby annotations |
| <rt> | Defines an explanation/pronunciation of characters (for East Asian typography) |
| <ruby> | Defines a ruby annotation (for East Asian typography) |
| <section> | Defines a section in the document |
| <summary> | Defines a visible heading for a <details> element |
| <time> | Defines a date/time |
| <wbr> | Defines a possible line-break |

New Input Types

|  |  |
| --- | --- |
| **New Input Types** | **New Input Attributes** |
| colordatedatetimedatetime-localemailmonthnumberrangesearchteltimeurlweek | Autocompleteautofocusformformactionformenctypeformmethodformnovalidateformtargetheight and widthlistmin and maxmultiplepattern (regexp)placeholderrequiredstep |

HTML5 Graphics

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <canvas> | Defines graphic drawing using JavaScript |
| <svg> | Defines graphic drawing using SVG |

New Media Elements

|  |  |
| --- | --- |
| **Tag** | **Description** |
| <audio> | Defines sound or music content |
| <embed> | Defines containers for external applications (like plug-ins) |
| <source> | Defines sources for <video> and <audio> |
| <track> | Defines tracks for <video> and <audio> |
| <video> | Defines video or movie content |

HTML5 Browser Support

HTML5 is supported in all modern browsers.

Define HTML5 Elements as Block Elements

* HTML5 defines eight new semantic HTML elements. All these are block-level elements.
* header, section, footer, aside, nav, main, article, figure
* You can also add any new element to HTML

Example:-

<html>

<head>

<script type="text/javascript">

document.createElement("myHero");

document.createElement("pankaj");

document.createElement("bipin");

</script>

 <style>

 myHero {

 display: block;

 background-color: #ddd;

 padding: 50px;

 font-size: 30px;

 }

 </style>

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

<myhero>My First Hero</myhero>

<pankaj>pankaj</pankaj>

<bipin>Bipin</bipin>

</body></html>

What are Semantic Elements?

* A semantic element clearly describes its meaning to both the browser and the developer.
* Examples of non-semantic elements: DIV tag and SPAN tag- Tells nothing about its content.
* Examples of semantic elements: FORM tag, TABLE tag, and IMG tag - Clearly defines its content.



1. **Advanced Input Type**

New Input types are as below



Example :

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

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>Untitled Document</title>

<style type="text/css">

.field-wrpr {

 float: left;

 width: 100%;

}

</style>

</head>

<body>

<form>

 <div class="field-wrpr">

 <h2> type Search</h2>

 <label> Search Website:

 <input type="search" name="mysearch">

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type Range</h2>

 <label> Select Number:

 <input type="range" value="1" min="1" max="10" step="0.5" name="mynumber">

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type Date Time</h2>

 <label> Date &amp; Time:

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

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type Email</h2>

 <label>

 Email Address: <input type="email" required="">

 <button type="submit">Save</button>

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type Date</h2>

 <label>

 Select Date: <input type="date" name="mydate">

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type Local Date &amp; Time</h2>

 <label>

 Local Date &amp; Time: <input type="datetime-local" name="mylocaldatetime">

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type URL</h2>

 <label>

 Website URL: <input type="url" name="mywebsite" required="">

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type URL</h2>

 <label>

 Select Month: <input type="month" name="mymonth">

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type Color</h2>

 <label>

 Select Color: <input type="color" name="mycolor">

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type tel</h2>

 <label>

 Telephone Number: <input type="tel" name="mytelephone" required="">

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type Week</h2>

 <label>

 Select Week: <input type="week" name="myweek">

 </label>

 </div>

 <div class="field-wrpr">

 <h2> type Number</h2>

 <label> Select Number:

 <input type="number" value="6" min="1.5" max="10" step="1.5" name="mynumber">

 </label>

 <p><strong>Note</strong>: If you try to enter the number out of the range (1-10) or text character it will show error.</p>

 </div>

</form>

</body></html>

1. **SVG**

What is SVG?

* SVG stands for Scalable Vector Graphics
* SVG is used to define vector-based graphics for the Web
* SVG defines the graphics in XML format
* SVG graphics do NOT lose any quality if they are zoomed or resized
* Every element and every attribute in SVG files can be animated
* SVG is a W3C recommendation
* SVG integrates with other W3C standards such as the DOM and XSL

SVG Advantages

* SVG images can be created and edited with any text editor
* SVG images can be searched, indexed, scripted, and compressed
* SVG images are scalable
* SVG images can be printed with high quality at any resolution
* SVG images are zoom able (and the image can be zoomed without degradation)
* SVG is an open standard
* SVG files are pure XML

SVG Example

* Start With svg
* Line Using SVG
* Rectangle Using SVG
* Circle Using SVG
* Text Using SVG
* Transform Using SVG
1. Start With svg

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Embedding SVG Into HTML Pages</title>

<style type="text/css">

svg{border:1px solid #000}

</style>

</head>

<body>

 <svg width="600" height="200">

 <text x="10" y="20" style="font-size:14px; color:#ccc">

 Your browser support SVG.

 </text>

 Sorry, your browser does not support SVG.

 </svg>

</body>

</html>

1. Line Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Create a Line with HTML5 SVG</title>

<style type="text/css">

 svg {border: 1px solid black; }

</style>

</head>

<body>

 <svg width="300" height="200">

 <line x1="10" y1="10" x2="150" y2="100" style="stroke:#999; stroke-width:1;"></line>

 </svg>

</body></html>

1. Rectangle Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Create a Rectangle with HTML5 SVG</title>

<style type="text/css">

 svg {border: 1px solid black;}

</style>

</head>

<body>

 <svg width="300" height="200">

 <rect x="50" y="50" width="200" height="100" style="fill:red; stroke:yellow; stroke-width:5;"></rect>

 </svg>

</body></html>

1. Circle Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Create a Circle with HTML5 SVG</title>

<style type="text/css">

 svg {border: 1px solid black;}

</style>

</head>

<body>

 <svg width="300" height="200">

 <circle cx="150" cy="100" r="100" style="fill:green; stroke:red; stroke-width:1;"></circle>

 </svg>

</body></html>

1. Text Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Render Text with HTML5 SVG</title>

<style type="text/css">

 svg {border: 1px solid black; }

</style>

</head>

<body>

 <svg width="300" height="200">

 <text x="20" y="30" style="fill:purple; font-size:22px; transform:rotate(10deg);">

 Welcome to Our Website!

 </text>

 <text x="20" y="30" dx="0" dy="20" style="fill:navy; font-size:14px; transform:rotate(45deg);">

 Here you will find lots of useful information.

 </text>

 </svg>

</body></html>

1. Transform Using SVG

<html lang="en"><head>

<meta charset="UTF-8">

<title>Rotate and Render Text with HTML5 SVG</title>

<style type="text/css">

 svg {border: 1px solid black;}

</style>

</head>

<body>

 <svg width="300" height="200">

 <text x="30" y="15" style="fill:purple; font-size:22px; transform:rotate(45deg);">

 <tspan style="fill:purple; font-size:22px;">

 Welcome to Our Website!

 </tspan>

 <tspan dx="-230" dy="20" style="fill:navy; font-size:14px;">

 Here you will find lots of useful information.

 </tspan>

 </text>

 </svg>

</body></html>

1. **Canvas Tag**

****

What is HTML Canvas?

* The HTML CANVAS element is used to draw graphics, on the fly, via scripting (usually JavaScript).
* The CANVAS element is only a container for graphics. You must use a script to actually draw the graphics.
* Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Example:

* Inserting CANVAS tag
* Line Using Canvas
* ARC Using Canvas
* Rectangle Using canvas
* Stroke Using canvas
* Cap Stroke Using canvas
* Inside a rectangle
* Inside a circle
1. Inserting CANVAS tag

<html lang="en"><head>

<meta charset="UTF-8">

<title>HTML5 Canvas</title>

<script type="text/javascript">

 window.onload = function(){

 var canvas = document.getElementById("myCanvas");

 // draw stuff here

 };

</script>

<style>

canvas{border:1px solid #000}

</style>

</head>

<body>

 <canvas id="myCanvas" width="300" height="200"></canvas>

</body></html>

1. Line Using Canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Drawing a Line on Canvas</title>

<style type="text/css">

 canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

 window.onload = function(){

 var canvas = document.getElementById("myCanvas");

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

 context.moveTo(20, 20);

 context.lineTo(100, 150);

 context.lineTo(300, 300);

 context.lineTo(75, 100);

 context.stroke();

 };

</script>

</head>

<body>

 <canvas id="myCanvas" width="700" height="400"></canvas>

</body></html>

1. ARC Using Canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Drawing an Arc on Canvas</title>

<style type="text/css">

 canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

 window.onload = function(){

 var canvas = document.getElementById("myCanvas");

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

 //context.arc(centerX, centerY, radius, startingAngle, endingAngle, counterclockwise);

 context.arc(150, 150, 50, 0 \* Math.PI, 0.5 \* Math.PI, true);

 context.stroke();

 };

</script>

</head>

<body>

 <canvas id="myCanvas" width="500" height="500"></canvas>

</body></html>

1. Rectangle Using canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Drawing a Rectangle on Canvas</title>

<style type="text/css">

 canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

 window.onload = function(){

 var canvas = document.getElementById("myCanvas");

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

 context.lineWidth = 2;

 context.strokeStyle = "red";

 //context.rect(x, y, width, height);

 context.rect(50, 50, 200, 200);

 context.stroke();

 };

</script>

</head>

<body>

 <canvas id="myCanvas" width="300" height="300"></canvas>

</body></html>

1. Stroke Using canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Setting Stroke Color and Width</title>

<style type="text/css">

 canvas{border: 1px solid #000; }

</style>

<script type="text/javascript">

 window.onload = function(){

 var canvas = document.getElementById("myCanvas");

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

 context.lineWidth = 5;

 context.strokeStyle = "blue";

 context.moveTo(50, 150);

 context.lineTo(250, 50);

 context.stroke();

 context.lineCap = "round";

 };

</script>

</head>

<body>

 <canvas id="myCanvas" width="300" height="200"></canvas>

</body></html>

1. Cap Stroke Using canvas

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Setting Stroke Cap Style</title>

<style type="text/css">

 canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

 window.onload = function(){

 var canvas = document.getElementById("myCanvas");

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

 context.lineWidth = 10;

 context.strokeStyle = "orange";

 //butt, round, and square.

 context.lineCap = "butt";

 context.arc(150, 150, 80, 1.2 \* Math.PI, 1.8 \* Math.PI, false);

 context.stroke();

 };

</script>

</head>

<body>

 <canvas id="myCanvas" width="300" height="200"></canvas>

</body></html>

1. Inside a rectangle

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Filling Color inside a Rectangle</title>

<style type="text/css">

 canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

 window.onload = function(){

 var canvas = document.getElementById("myCanvas");

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

 context.rect(50, 50, 200, 100);

 context.fillStyle = "#999";

//It is recommended to use the fill() method before the stroke() method in order to render the stroke correctly.

 context.fill();

 context.lineWidth = 5;

 context.strokeStyle = "blue";

 context.stroke();

 };

</script>

</head>

<body>

 <canvas id="myCanvas" width="300" height="200"></canvas>

</body></html>

1. Inside a circle

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Filling Color inside a Circle</title>

<style type="text/css">

 canvas{border: 1px solid #000;}

</style>

<script type="text/javascript">

 window.onload = function(){

 var canvas = document.getElementById("myCanvas");

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

 context.arc(150, 100, 70, 0, 2 \* Math.PI, false);

 context.fillStyle = "#FB8B89";

 //It is recommended to use the fill() method before the stroke() method in order to render the stroke correctly.

 context.fill();

 context.lineWidth = 5;

 context.strokeStyle = "black";

 context.stroke();

 };

</script>

</head>

<body>

<canvas id="myCanvas" width="300" height="200"></canvas>

 </body></html>

**5) Audio Tag**

* The audio tag defines sound, such as music or other audio streams.
* Currently, there are 3 supported file formats for the audio tag element: MP3, Wav, and Ogg

Example :

<html lang="en"><head>

 <meta charset="UTF-8">

 <title>The HTML5 audio Element</title>

</head>

<body>

 <audio control autoplay="true" src="6.MUSIC/Kalimba.mp3” >

 Your browser does not support the HTML5 audio element.

 </audio>

</body></html>

**5) Video Tag**

* Before HTML5, there was no standard for showing videos on a web page.
* Before HTML5, videos could only be played with a plug-in (like flash).
* The HTML5 video tag element specifies a standard way to embed a video in a web page.

Example :

<html lang="en"><head>

 <meta charset="UTF-8">

 <title>The HTML5 video element</title>

<style>

video{

 position:relative;

}

.video-text p{

 position:relative;top:100px; font-size:30px;font-weight:bold;color:#fff

}

</style></head>

<body>

 <div class="video-text">

 <video autoplay="" controls="" src="7.VIDEO/video.mkv">

 Your browser does not support the HTML5 Video element.

 </video>

 <p>Hello</p>

</div>

</body></html>

1. **Geolocation**

The HTML5 geolocation feature lets you find out the geographic coordinates (latitude and longitude numbers) of the current location of your website's visitor. This feature is helpful for providing better browsing experience to the site visitor. For example, you can return the search results that are physically close to the user's location.

Example:

* GETTING CO-ORDINATES
* HANDELING ERROR AND REJECTION
* SHOW GEOLOCATION AS IMAGE
* GELOCATION BY GOOGLE MAP
* CURRENT LOCATION OF VISITOR
1. GETTING CO-ORDINATES

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of HTML5 Geolocation</title>

<script type="text/javascript">

 function showPosition(){

 if(navigator.geolocation){

 navigator.geolocation.getCurrentPosition(function(position){

 var positionInfo = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";

 document.getElementById("result").innerHTML = positionInfo;

 });

 } else{

 alert("Sorry, your browser does not support HTML5 geolocation.");

 }

 }

</script>

</head>

<body>

 <div id="result">

 <!--Position information will be inserted here-->

 </div>

 <button type="button" onclick="showPosition();">Show Position</button>

</body></html>

1. HANDELING ERROR AND REJECTION

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Handling Geolocation Errors and Rejections</title>

<script type="text/javascript">

 // Set up global variable

 var result;

 function showPosition(){

 // Store the element where the page displays the result

 result = document.getElementById("result");

 // If geolocation is available, try to get the visitor's position

 if(navigator.geolocation){

 navigator.geolocation.getCurrentPosition(successCallback, errorCallback);

 result.innerHTML = "Getting the position information...";

 } else{

 alert("Sorry, your browser does not support HTML5 geolocation.");

 }

 };

 // Define callback function for successful attempt

 function successCallback(position){

 result.innerHTML = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";

 }

 // Define callback function for failed attempt

 function errorCallback(error){

 if(error.code == 1){

 result.innerHTML = "You've decided not to share your position, but it's OK. We won't ask you again.";

 } else if(error.code == 2){

 result.innerHTML = "The network is down or the positioning service can't be reached.";

 } else if(error.code == 3){

 result.innerHTML = "The attempt timed out before it could get the location data.";

 } else{

 result.innerHTML = "Geolocation failed due to unknown error.";

 }

 }

</script>

</head>

<body>

 <div id="result">

 <!--Position information will be inserted here-->

 </div>

 <button type="button" onclick="showPosition();">Show Position</button>

</body></html>

1. SHOW GEOLOCATION AS IMAGE

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Showing Geolocation on Google Map</title>

<script type="text/javascript">

 function showPosition(){

 navigator.geolocation.getCurrentPosition(showMap);

 }

 function showMap(position){

 // Get location data

 var latlong = position.coords.latitude + "," + position.coords.longitude;

 // Set Google map source url

 var mapLink = "http://maps.googleapis.com/maps/api/staticmap?center="+latlong+"&zoom=16&size=400x300&output=embed";

 // Create and insert Google map

 document.getElementById("embedMap").innerHTML = "<img alt='Map Holder' src='"+ mapLink +"'>";

 }

</script>

</head>

<body>

 <button type="button" onclick="showPosition();">Show My Position on Google Map</button>

 <div id="embedMap">

 <!--Google map will be embedded here-->

 </div>

</body></html>

1. GELOCATION BY GOOGLE MAP

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Showing Location on Google Map</title>

<script src="http://maps.google.com/maps/api/js?sensor=false"></script>

<script type="text/javascript">

function showPosition(){

 if(navigator.geolocation){

 navigator.geolocation.getCurrentPosition(showMap, showError);

 } else{

 alert("Sorry, your browser does not support HTML5 geolocation.");

 }

}

// Define callback function for successful attempt

function showMap(position){

 // Get location data

 lat = position.coords.latitude;

 long = position.coords.longitude;

 var latlong = new google.maps.LatLng(lat, long);

 var myOptions = {

 center: latlong,

 zoom: 16,

 mapTypeControl: true,

 navigationControlOptions: {style:google.maps.NavigationControlStyle.SMALL}

 }

 var map = new google.maps.Map(document.getElementById("embedMap"), myOptions);

 var marker = new google.maps.Marker({position:latlong, map:map, title:"You are here!"});

}

// Define callback function for failed attempt

function showError(error){

 if(error.code == 1){

 result.innerHTML = "You've decided not to share your position, but it's OK. We won't ask you again.";

 } else if(error.code == 2){

 result.innerHTML = "The network is down or the positioning service can't be reached.";

 } else if(error.code == 3){

 result.innerHTML = "The attempt timed out before it could get the location data.";

 } else{

 result.innerHTML = "Geolocation failed due to unknown error.";

 }

}

</script>

</head>

<body>

 <button type="button" onclick="showPosition();">Show My Position on Google Map</button>

 <div id="embedMap" style="width: 400px; height: 300px;">

 <!--Google map will be embedded here-->

 </div>

</body></html>

1. CURRENT LOCATION OF VISITOR

<html lang="en"><head>

<meta charset="UTF-8">

<title>Example of Getting Current Position</title>

<script type="text/javascript">

 // Set global variable

 var watchID;

 function showPosition(){

 if(navigator.geolocation){

 watchID = navigator.geolocation.watchPosition(successCallback);

 } else{

 alert("Sorry, your browser does not support HTML5 geolocation.");

 }

 }

 function successCallback(position){

 toggleWatchBtn.innerHTML = "Stop Watching";

 // Check position has been changed or not before doing anything

 if(prevLat != position.coords.latitude || prevLong != position.coords.longitude){

 // Set previous location

 var prevLat = position.coords.latitude;

 var prevLong = position.coords.longitude;

 // Get current position

 var positionInfo = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";

 document.getElementById("result").innerHTML = positionInfo;

 }

 }

 function startWatch(){

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

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

 toggleWatchBtn.onclick = function(){

 if(watchID){

 toggleWatchBtn.innerHTML = "Start Watching";

 navigator.geolocation.clearWatch(watchID);

 watchID = false;

 }

 else{

 toggleWatchBtn.innerHTML = "Aquiring Geo Location...";

 showPosition();

 }

 }

 }

 // Initialise the whole system (above)

 window.onload = startWatch;

</script>

</head>

<body>

 <button type="button" id="toggleWatchBtn">Start Watching</button>

 <div id="result">

 <!--Position information will be inserted here-->

 </div>

</body></html>