HTML5

What is HTML5?

HTML5 is the latest standard for HTML.

The previous version of HTML, HTML 4.01, came in 1999, and the internet has changed significantly since then.

HTML5 was designed to replace both HTML 4, XHTML, and the HTML DOM Level 2.

It was specially designed to deliver rich content without the need for additional plugins. The current version delivers everything from animation to graphics, music to movies, and can also be used to build complicated web applications.

HTML5 is also cross-platform. It is designed to work whether you are using a PC, or a Tablet, a Smartphone, or a Smart TV.

How Did HTML5 Get Started?

HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).

WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.

Some rules for HTML5 were established:

- New features should be based on HTML, CSS, DOM, and JavaScript
- The need for external plugins (like Flash) should be reduced
- Error handling should be easier than in previous versions
- Scripting has to be replaced by more markup
- HTML5 should be device-independent
- The development process should be visible to the public

The HTML5 <!DOCTYPE>

In HTML5 there is only one DOCTYPE declaration, and it is very simple:

<!DOCTYPE html>

A Minimum HTML5 Document

Below is a simple HTML5 document, with the minimum of required tags:

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

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

</html>
```

HTML5 - New Features

Some of the most interesting new features in HTML5 are:

- The <canvas> element for 2D drawing
- The <video> and <audio> elements for media playback
- Support for local storage
- New content-specific elements, like <article>, <footer>, <header>, <nav>, <section>
- New form controls, like calendar, date, time, email, url, search

Browser Support for HTML5

All major browsers (Chrome, Firefox, Internet Explorer, Safari, Opera) support the new HTML5 elements and APIs, and continue to add new HTML5 features to their latest versions.

The HTML 5 working group includes AOL, Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera, and hundreds of other vendors.

New Elements in HTML5

The internet, and the use of the internet, has changed a lot since 1999, when HTML 4.01 became a standard.

Today, several elements in HTML 4.01 are obsolete, never used, or not used the way they were intended. All those elements are removed or re-written in HTML5.

To better handle today's internet needs, HTML5 has also included new elements for drawing graphics, displaying media content, for better page structure and better form handling, and several new APIs, such as drag and drop, get the geographical position of a user, store local data, and more.

Below is a list of the new HTML elements, introduced by HTML5, and a description of what they are used for.

The New <canvas> Element

Note: The links in the tables below point to our HTML5 Reference. However, you will learn more about these new elements in this tutorial.

Tag	Description
<canvas>	Defines graphic drawing using JavaScript

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

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

New Semantic/Structural Elements

HTML5 offers new elements for better structure.

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 outside it
<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

Removed Elements

The following HTML 4.01 elements has been removed from HTML5:

- <acronym>
- <applet>
- <basefont>
- <big>
-
- <dir>
- <font>
- <frame>
- <frameset>
- <noframes>
- <strike>
- <tt>

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: and - Tells nothing about its content.

Examples of **semantic** elements: <form>, <table>, and <img> - Clearly defines its content.

Browser Support

Internet Explorer 9+, Firefox, Chrome, Safari and Opera supports the semantic elements described in this chapter.

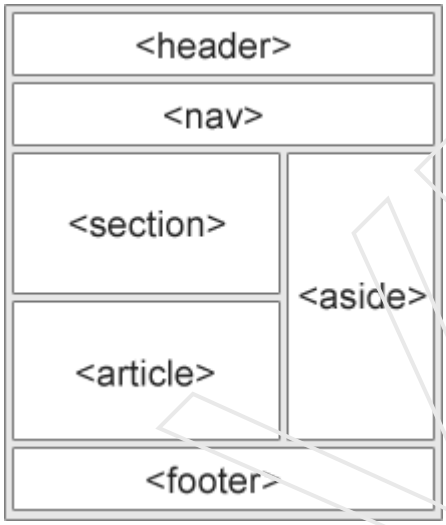
Note: Internet Explorer 8 and earlier does not support these elements. However, there is a solution. Look at the end of this chapter.

New Semantic Elements in HTML5

Many of existing web sites today contains HTML code like this: , , or , to indicate navigation links, header, and footer.

HTML5 offers new semantic elements to clearly define different parts of a web page:

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



HTML5 <section> Element

The <section> element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

Example

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

HTML5 <article> Element

The <article> element specifies independent, self-contained content.

An article should make sense on its own and it should be possible to distribute it independently from the rest of the web site.

Examples of where an <article> element can be used:

- Forum post
- Blog post
- News story
- Comment

Example

```
<article>
  <h1>Internet Explorer 9</h1>
  <p>Windows Internet Explorer 9 (abbreviated as IE9) was released to
  the  public on March 14, 2011 at 21:00 PDT.....</p>
</article>
```

HTML5 <nav> Element

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!

Example

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

HTML5 <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.

Example

```
<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>
```

HTML5 <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.

You can have several <header> elements in one document.

The following example defines a header for an article:

Example

```
<article>
  <header>
    <h1>Internet Explorer 9</h1>
    <p><time pubdate datetime="2011-03-15"></time></p>
  </header>
  <p>Windows Internet Explorer 9 (abbreviated as IE9) was released to
  the  public on March 14, 2011 at 21:00 PDT.....</p>
</article>
```

HTML5 <footer> Element

The <footer> element specifies a footer for a document or section.

A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You can have several <footer> elements in one document.

Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p><time pubdate datetime="2012-03-01"></time></p>
</footer>
```

HTML5 <figure> and <figcaption> Elements

The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

While the content of the <figure> element is related to the main flow, its position is independent of the main flow, and if removed it should not affect the flow of the document.

The <figcaption> tag defines a caption for a <figure> element.

The <figcaption> element can be placed as the first or last child of the <figure> element.

Example

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

HTML5 New Input Types

HTML5 has several new input types for forms. These new features allow better input control and validation. This chapter covers the new input types:

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

Note: Not all browsers support all the new input types. However, you can already start using them; If they are not supported, they will behave as regular text fields.

Input Type: color

The color type is used for input fields that should contain a color.

Example

Select a color from a color picker:

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

Input Type: date

The date type allows the user to select a date.

Example

Define a date control:

Birthday: <input type="date" name="bday">

Input Type: datetime

The datetime type allows the user to select a date and time (with time zone).

Example

Define a date and time control (with time zone):

Birthday (date and time): <input type="datetime" name="bdaytime">

Input Type: datetime-local

The datetime-local type allows the user to select a date and time (no time zone).

Example

Define a date and time control (no time zone):

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

Input Type: email

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

Example

Define a field for an e-mail address (will be automatically validated when submitted):

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

Tip: Safari on iPhone recognizes the email type, and changes the on-screen keyboard to match it (adds @ and .com options).

Input Type: month

The month type allows the user to select a month and year.

Example

Define a month and year control (no time zone):

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

Input Type: number

The number type is used for input fields that should contain a numeric value.

You can also set restrictions on what numbers are accepted:

Example

Define a numeric field (with restrictions):

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

Use the following attributes to specify restrictions:

- max - specifies the maximum value allowed
- min - specifies the minimum value allowed
- step - specifies the legal number intervals
- value - Specifies the default value

Input Type: range

The range type is used for input fields that should contain a value from a range of numbers.

You can also set restrictions on what numbers are accepted.

Example

Define a control for entering a number whose exact value is not important (like a slider control):

<input type="range" name="points" min="1" max="10">

Use the following attributes to specify restrictions:

- max - specifies the maximum value allowed
- min - specifies the minimum value allowed
- step - specifies the legal number intervals
- value - Specifies the default value

Input Type: search

The search type is used for search fields (a search field behaves like a regular text field).

Example

Define a search field (like a site search, or Google search):

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

Input Type: tel

The tel type is used for input fields that should contain a telephone number.

Example

Define a field for entering a telephone number:

Telephone: <input type="tel" name="usrtel">

Input Type: time

The time type allows the user to select a time.

Example

Define a control for entering a time (no time zone):

Select a time: <input type="time" name="usr_time">

Input Type: url

The url type is used for input fields that should contain a URL address.

The value of the url field is automatically validated when the form is submitted.

Example

Define a field for entering a URL: Add your homepage: <input type="url" name="homepage">
Tip: Safari on iPhone recognizes the url input type, and changes the on-screen keyboard to match it (adds .com option).

Input Type: week

The week type allows the user to select a week and year.

Example

Define a week and year control (no time zone):

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

HTML5 <input> Tag

Tag	Description
<input>	Defines an input control

HTML5 New Form Attributes

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

New attributes for <form>:

- autocomplete
- novalidate

New attributes for <input>:

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

<form> / <input> autocomplete Attribute

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

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

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

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

Example

An HTML form with autocomplete on (and off for one input field):

```
<form action="demo_form.asp" 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>
```

Tip: In some browsers you may need to activate the autocomplete function for this to work.

<form> novalidate Attribute

The novalidate attribute is a boolean attribute.

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

Example

Indicates that the form is not to be validated on submit:

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

<input> autofocus Attribute

The autofocus attribute is a boolean attribute.

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

Example

Let the "First name" input field automatically get focus when the page loads:

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

<input> form Attribute

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

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

Example

An input field located outside the HTML form (but still a part of the form):

```
<form action="demo_form.asp" 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">
```

<input> formaction Attribute

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

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

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

Example

An HTML form with two submit buttons, with different actions:

```
<form action="demo_form.asp">
  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="demo_admin.asp"
  value="Submit as admin">
</form>
```

<input> formenctype Attribute

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

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

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

Example

Send form-data that is default encoded (the first submit button), and encoded as "multipart/form-data" (the second submit button):

```
<form action="demo_post_enctype.asp" method="post">
  First name: <input type="text" name="fname"><br>
```

```
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```

<input> formmethod Attribute

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

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

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

Example

The second submit button overrides the HTTP method of the form:

```
<form action="demo_form.asp" 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="demo_post.asp"
  value="Submit using POST">
</form>
```

<input> formnovalidate Attribute

The novalidate attribute is a boolean attribute.

When present, it specifies that the <input> element should not be validated when submitted.

The formnovalidate attribute overrides the novalidate attribute of the <form> element.

Note: The formnovalidate attribute can be used with type="submit".

Example A form with two submit buttons (with and without validation):

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

<input> 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.

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

Example A form with two submit buttons, with different target windows:

```
<form action="demo_form.asp">
  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>
```

<input> height and width Attributes

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

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

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

Example Define an image as the submit button, with height and width attributes:

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

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

Example

An <input> element with pre-defined values in a <datalist>:

```
<input list="browsers">

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

<input> min and max Attributes

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

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

Example

<input> elements with min and max values:

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">

<input> multiple Attribute

The multiple attribute is a boolean attribute.

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

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

Example

A file upload field that accepts multiple values:

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

<input> pattern Attribute

The pattern attribute specifies a regular expression that the <input> element's value is checked against.

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

Tip: Use the global title attribute to describe the pattern to help the user.

Example

An input field that can contain only three letters (no numbers or special characters):

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

<input> placeholder Attribute

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

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

Example

An input field with a placeholder text:

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

<input> required Attribute

The required attribute is a boolean attribute.

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

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

Example

A required input field:

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

<input> 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.

Tip: The step attribute can be used together with the max and min attributes to create a range of legal values.

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

Example

An input field with a specified legal number intervals:

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

HTML5 <input> Tag

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control

What is Canvas?

The HTML5 <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.

Browser Support

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <canvas> element.

Note: Internet Explorer 8 and earlier versions, do not support the <canvas> element.

Create a Canvas

A canvas is a rectangular area on an HTML page, and it is specified with the <canvas> element.

Note: By default, the <canvas> element has no border and no content.

The markup looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Note: 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.

Tip: You can have multiple <canvas> elements on one HTML page. To add a border, use the style attribute:

Example

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

Draw Onto The Canvas With JavaScript

All drawing on the canvas must be done inside a JavaScript:

Example

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
</script>
```

Example explained:

First, find the <canvas> element:

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

Then, call its getContext() method (you must pass the string "2d" to the getContext() method):

var ctx = c.getContext("2d");

The getContext("2d") object is a built-in HTML5 object, with many properties and methods for drawing paths, boxes, circles, text, images, and more.

The next two lines draw a red rectangle:

```
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
```

The fillStyle property can be a CSS color, a gradient, or a pattern. The default fillStyle is #000000 (black).

The fillRect(*x,y,width,height*) method draws a rectangle filled with the current fill style.

Canvas Coordinates

The canvas is a two-dimensional grid.

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

So, the fillRect() method above had the parameters (0,0,150,75).

This means: Start at the upper-left corner (0,0) and draw a 150x75 pixels rectangle.

Coordinates Example

Mouse over the rectangle below to see its x and y coordinates:

X

Y

Canvas - Paths

To draw straight lines on a canvas, we will use the following two methods:

- moveTo(*x,y*) defines the starting point of the line
- lineTo(*x,y*) defines the ending point of the line

To actually draw the line, we must use one of the "ink" methods, like stroke().

Example

Define a starting point in position (0,0), and an ending point in position (200,100). Then use the stroke() method to actually draw the line:

JavaScript:

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

To draw a circle on a canvas, we will use the following method:

- arc(x,y,r,start,stop)

To actually draw the circle, we must use one of the "ink" methods, like stroke() or fill().

Example

Create a circle with the arc() method:

JavaScript:

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

Canvas - Text

To draw text on a canvas, the most important property and methods are:

- font - defines the font properties for text
- fillText(*text*,*x*,*y*) - Draws "filled" text on the canvas
- strokeText(*text*,*x*,*y*) - Draws text on the canvas (no fill)

Using fillText():

Example

Write a 30px high filled text on the canvas, using the font "Arial":

JavaScript:

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

Using strokeText():

Example Write a 30px high text (no fill) on the canvas, using the font "Arial":

JavaScript:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("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, and then draw the shape, like a rectangle, text, or a line.

Using createLinearGradient():

Example Create a linear gradient. Fill rectangle with the gradient:

JavaScript:

```
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);
```

Using createRadialGradient():

Example

Create a radial/circular gradient. Fill rectangle with the gradient:

JavaScript:

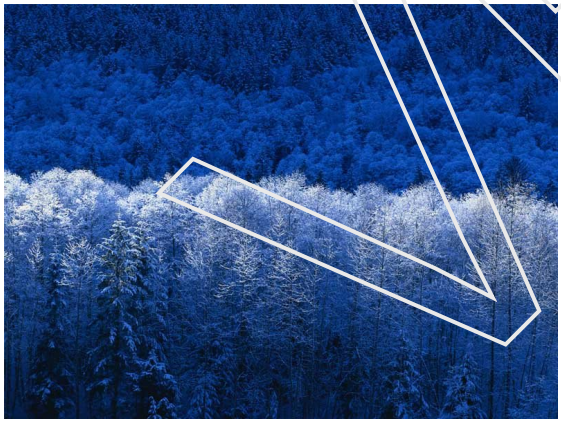
```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// 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);
```

Canvas - Images

To draw an image on a canvas, we will use the following method:

- drawImage(*image,x,y*)

Image to use:



Example

Draw the image onto the canvas:

JavaScript:

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img,10,10);
```

The HTML <canvas> Tag

Tag	Description
<canvas>	Used to draw graphics, on the fly, via scripting (usually JavaScript)

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 Advantages

Advantages of using SVG over other image formats (like JPEG and GIF) are:

- 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 zoomable (and the image can be zoomed without degradation)

Browser Support

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support inline SVG.

Embed SVG Directly Into HTML Pages

In HTML5, you can embed SVG elements directly into your HTML page:

Example

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

<svg width="300" height="200">
  <polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>

</body>
</html>
```

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

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

Canvas	SVG
• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games	• Resolution independent • Support for event handlers • Best suited for applications with large rendering areas (Google Maps) • Slow rendering if complex (anything that uses the DOM a lot will be slow) • Not suited for game applications

Video on the Web

Before HTML5, there was no standard for showing videos/movies on web pages.

Before HTML5, videos could only be played with a plug-in (like flash). However, different browsers supported different plug-ins.

HTML5 defines a new element which specifies a standard way to embed a video or movie on a web page: the <video> element.

Browser Support

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <video> element.

Note: Internet Explorer 8 and earlier versions, do not support the <video> element.

HTML5 Video - How It Works To show a video in HTML5, this is all you need:

Example

```
<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>
```

The control attribute adds video controls, like play, pause, and volume.

It is also a good idea to always include width and height attributes. If height and width are set, the space required for the video is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the video, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the video loads).

You should also insert text content between the <video> and </video> tags for browsers that do not support the <video> element.

The <video> element allows multiple <source> elements. <source> elements can link to different video files. The browser will use the first recognized format.

Video Formats and Browser Support

Currently, there are 3 supported video formats for the <video> element: MP4, WebM, and Ogg:

- MP4 = MPEG 4 files with H264 video codec and AAC audio codec
- WebM = WebM files with VP8 video codec and Vorbis audio codec
- Ogg = Ogg files with Theora video codec and Vorbis audio codec

MIME Types for Video Formats

Format	MIME-type
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

HTML5 <video> - DOM Methods and Properties

HTML5 has DOM methods, properties, and events for the <video> and <audio> elements.

These methods, properties, and events allow you to manipulate <video> and <audio> elements using JavaScript.

There are methods for playing, pausing, and loading, for example and there are properties (like duration and volume). There are also DOM events that can notify you when the <video> element begins to play, is paused, is ended, etc.

HTML5 Video Tags

Tag	Description
<video>	Defines a video or movie
<source>	Defines multiple media resources for media elements, such as <video> and <audio>
<track>	Defines text tracks in media players

Audio on the Web

Before HTML5, there was no standard for playing audio files on a web page.

Before HTML5, audio files had to be played with a plug-in (like flash). However, different browsers supported different plug-ins.

HTML5 defines a new element which specifies a standard way to embed an audio file on a web page: the <audio> element.

Browser Support

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <audio> element.

Note: Internet Explorer 8 and earlier versions, do not support the <audio> element.

HTML5 Audio - How It Works

To play an audio file in HTML5, this is all you need:

Example

```
<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>
```

The control attribute adds audio controls, like play, pause, and volume.

You should also insert text content between the <audio> and </audio> tags for browsers that do not support the <audio> element.

The <audio> element allows multiple <source> elements. <source> elements can link to different audio files. The browser will use the first recognized format.

Audio Formats and Browser Support

Currently, there are 3 supported file formats for the <audio> element: MP3, Wav, and Ogg:

MIME Types for Audio Formats

Format	MIME-type
MP3	audio/mpeg
Ogg	audio/ogg
Wav	audio/wav

HTML5 Audio Tags

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements, such as <video> and <audio>

What is Multimedia?

Multimedia comes in many different formats. It can be almost anything you can hear or see.

Examples: Pictures, music, sound, videos, records, films, animations, and more.

Modern web pages often have embedded multimedia elements, and modern browsers have support for various multimedia formats.

In this tutorial you will learn about the different multimedia formats.

Internet Browser Support

The first Internet browsers had support for text only, and even the text support was limited to a single font in a single color. Then came browsers with support for colors, fonts and text styles, and support for pictures was also added.

The support for sounds, animations, and videos is handled in different ways by various browsers. Some multimedia elements is supported, and some requires an extra helper program (a plug-in) to work.

You will learn more about plug-ins in the next chapters.

Multimedia Formats

Multimedia elements (like sounds or videos) are stored in media files.

The most common way to discover the type of a file, is to look at the file extension. When a browser sees the file extension .htm or .html, it will treat the file as an HTML file. The .xml extension indicates an XML file, and the .css extension indicates a style sheet file. Pictures are recognized by extensions like .gif, .png and .jpg.

Multimedia files also have their own formats and different extensions
like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

Video Formats

MP4 is the new and upcoming format for internet video.
MP4 is recommended by YouTube.
MP4 is supported by Flash players and HTML5.

HTML Helpers (Plug-ins)

A helper application is a small computer program that extends the standard functionality of the browser. Helper applications are also called plug-ins. Examples of well-known plug-ins are Java applets and Adobe Flash Player. Plug-ins can be added to web pages with the <object> tag or the <embed> tag. Plug-ins can be used for many purposes: to display maps, scan for viruses, verify your bank id, and much more. The restrictions are few.

What is The Best Way to Play Audio or Video in HTML?

The best way to embed audio in a web page is to use the HTML5 <audio> element.

The best way to embed video in a web page is to use the HTML5 <video> element.

The <object> Element

The <object> element is supported in all major browsers.

The <object> element defines an embedded object within an HTML document.

It is used to to embed plug-ins (like Java applets, ActiveX, PDF, and Flash) in web pages.

It can also be used to embed another webpage, or web content like images, into HTML documents.

The text between <object> and </object> is displayed if the browser doesn't support the tag.
The HTML <param> tag is used to pass parameters to the plug in.

Example

<object width="400" height="50" data="bookmark.swf"></object>

The <embed> Element

The <embed> element is supported in all major browsers.

The <embed> element defines a container for an external application or interactive content (a plug-in).

Many web browsers have supported the <embed> element for a long time. However, it has not been a part of the HTML specification before HTML5.

The <embed> element will validate in an HTML5 page, but not in an HTML 4 page.

Example

<embed width="400" height="50" src="bookmark.swf">

Note: that the <embed> element does not have a closing tag. It can not contain alternative text.

Playing a YouTube Video in HTML

If you want to play a video in a web page, you can upload the video to YouTube and insert the proper HTML code to display the video:

Example - YouTube iFrame

<iframe width="420" height="345"
src="http://www.youtube.com/embed/YGSx2_Czz4k">
</iframe>