2. HTML, XHTML & HTML5

Document metadata

The <head> element is a container for meta data (**data about data**).HTML meta data is data about the HTML document. Metadata is not displayed. Meta data typically define document title, styles, links, scripts, and other meta information.

The following tags describes meta data: <title>, <style>, <meta>, <link>, <script>, and <base>.

XHTML

- XHTML stands for E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
- XHTML is almost identical to HTML
- XHTML is stricter than HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers

Purpose of XHTML

Many pages on the internet contain "bad" HTML.This HTML code works fine in most browsers (even if it does not follow the HTML rules):

```
<html>
<head>
  <title>This is bad HTML</title>
<body>
  <h1>Bad HTML
  <p>This is a paragraph
</body>
```

Today's market consists of different browser technologies. Some browsers run on computers, and some browsers run on mobile phones or other small devices. Smaller devices often lack the resources or power to interpret "bad" markup.XML is a markup language where documents must be marked up correctly (be "well-formed").By combining the strengths of HTML and XML, XHTML was developed.XHTML is HTML redesigned as XML.

The Most Important Differences from HTML:

Document Structure

- XHTML DOCTYPE is **mandatory**
- The xmlns attribute in <html> is **mandatory**
- <html>, <head>, <title>, and <body> are **mandatory**

XHTML Elements

- XHTML elements must be **properly nested**
- XHTML elements must always be **closed**
- XHTML elements must be in **lowercase**
- XHTML documents must have **one root element**

XHTML Attributes

- Attribute names must be in **lower case**
- Attribute values must be **quoted**
- Attribute minimization is **forbidden**

<!DOCTYPE ....> Is Mandatory

An XHTML document must have an XHTML DOCTYPE declaration.

A complete list of all the XHTML Doctypes is found in our HTML Tags Reference.

The <html>, <head>, <title>, and <body> elements must also be present, and the xmlns attribute in <html> must specify the xml namespace for the document. This example shows an XHTML document with a minimum of required tags:<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

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

```
<head>
  <title>Title of document</title>
</head>

<body>
  some content
</body>

</html>
```

XHTML Rules::-

Elements Must Be Properly Nested

In HTML, some elements can be improperly nested within each other, like this:

```
<b><i>This text is bold and italic</b></i>
```

In XHTML, all elements must be properly nested within each other, like this:

```
<b><i>This text is bold and italic</i></b>
```

Elements Must Always Be Closed

This is wrong:

```
<p>This is a paragraph
<p>This is another paragraph
```

This is correct:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Empty Elements Must Also Be Closed

This is wrong:

```
A break: <br>
A horizontal rule: <hr>
An image: <img src="happy.gif" alt="Happy face">
```

This is correct:

```
A break: <br />
A horizontal rule: <hr />
An image: <img src="happy.gif" alt="Happy face" />
```

Elements Must Be In Lower Case

This is wrong:

```
<BODY>
<P>This is a paragraph</P>
</BODY>
```

This is correct:

```
<body>
<p>This is a paragraph</p>
</body>
```

Attribute Names Must Be In Lower Case

This is wrong:

```
<table WIDTH="100%">
```

This is correct:

```
<table width="100%">
```

Attribute Values Must Be Quoted

This is wrong:

```
<table width=100%>
```

This is correct:

```
<table width="100%">
```

Attribute Minimization Is Forbidden

Wrong:

<input type="checkbox" name="vehicle" value="car" checked />
Correct:
<input type="checkbox" name="vehicle" value="car" checked="checked" />
Wrong:
<input type="text" name="lastname" disabled />
Correct:
<input type="text" name="lastname" disabled="disabled" />

Steps For Converting HTML to XHTML

1. Add an XHTML <!DOCTYPE> to the first line of every page
2. Add an xmlns attribute to the html element of every page
3. Change all element names to lowercase
4. Close all empty elements
5. Change all attribute names to lowercase
6. Quote all attribute values

HTML5

The DOCTYPE declaration for HTML5 is very simple:
<!DOCTYPE html>
The character encoding (charset) declaration is also very simple:
<meta charset="UTF-8">

HTML5 Example:

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

The default character encoding in HTML5 is UTF-8.

HTML5 New elements:
The most interesting new elements are:
New **semantic** elements like <header>, <footer>, <article>, and <section>.
New form **controls** like number, date, time, calendar, and range.
New **graphic** elements: <svg> and <canvas>.
New **multimedia** elements: <audio> and <video>.

New HTML5 API's (Application Programming Interfaces)

The most interesting new API's are:

- HTML Geolocation
- HTML Drag and Drop
- HTML Local Storage
- HTML Application Cache

- HTML Web Workers
- HTML SSE

Local storage is a powerful replacement for cookies.

Tim Berners-Lee invented the "World Wide Web" in 1989, and the Internet took off in the 1990s.From 1991 to 1998, HTML developed from version 1 to version 4. In 2000, the World Wide Web Consortium (W3C) recommended XHTML 1.0. The XHTML syntax was strict, and the developers were forced to write valid and "well-formed" code. In 2004, WHATWG (the Web Hypertext Application Technology Group) was formed in response to slow W3C development, and W3C's decision to close down the development of HTML, in favor of XHTML. WHATWG wanted to develop HTML, consistent with how the web was used, while being backward compatible with older versions of HTML. In the period 2004-2006, the WHATWG initiative gained support by the major browser vendors. In 2006, W3C announced that they would support WHATWG. In 2008, the first HTML5 public draft was released In 2012, WHATWG and W3C decided on a separation:

HTML5 Browser Support

We can teach old browsers to handle HTML5

Define HTML5 Elements as Block Elements

HTML5 defines 8 new **semantic** HTML elements. All these are **block level** elements.

To secure correct behavior in older browsers, you can set the CSS **display** property to **block**:

Example

```
header, section, footer, aside, nav, main, article, figure {
  display: block;
}
```

HTML5 Semantic Elements

Semantics means (from Ancient Greek), is the study of meaning.Semantic elements are elements with a meaning.

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.

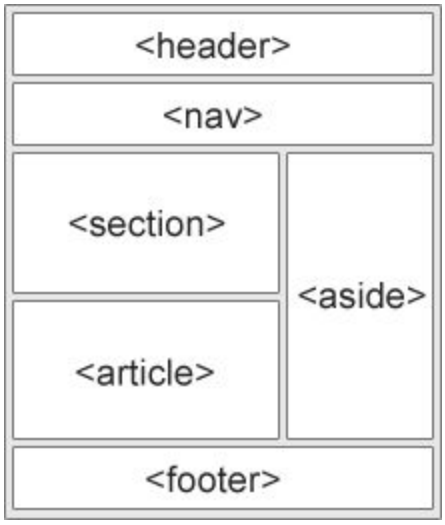
Examples of **semantic** elements: <form>, <table>, and <img> - Clearly defines its content.

New Semantic Elements in HTML5

Many web sites contain HTML code like:
to indicate navigation, header, and footer.

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

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



Various HTML5 Elements

<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.**" A Web site's home page could be split into sections for introduction, content, and contact information.
Example

```
<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

Example

```
<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>
```

Nesting Semantic Elements

In the HTML5 standard, the <article> element defines a complete, self-contained block of related elements. The <section> element is defined as a block of related elements. On the Internet, we will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <sections> elements. We will

also find pages with <section> elements containing <section> elements, and <article> elements containing <article> elements.

Newspaper: The sports **articles** in the sports **section**, have a technical **section** in each **article**.

<header> Element

The <header> element specifies a header for a document or section. <header> element should be used as a container for introductory content. We can have several <header> elements in one document. The following example defines a header for an article:

Example

```
<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. 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>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.
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>
```

<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>
```

<figure> and <figcaption> Elements

In books and newspapers, it is common to have captions with images.
The purpose of a caption is to add a visual explanation to an image.
With HTML5, images and captions can be grouped together in **<figure>** elements:

Example

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

The **<img>** element defines the image, the **<figcaption>** element defines the caption.

Why Semantic HTML5 Elements?

With HTML4, developers used their own favorite attribute names to style page elements:
header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, ...
This made it impossible for search engines to identify the correct web page content.
With HTML5 elements like: <header> <footer> <nav> <section> <article>, this will become easier.
According to the W3C, a Semantic Web:

"Allows data to be shared and reused across applications, enterprises, and communities."

Semantic Elements in HTML5

Below is an alphabetical list of the new semantic elements in HTML5.

Tag	Description

<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

HTML Migration

Migration from HTML4 to HTML5

This chapter is entirely about how to **migrate** from a typical **HTML4** page to a typical **HTML5** page.
This chapter demonstrates how to convert an existing HTML4 page into an HTML5 page, without destroying anything of the original content or structure.

You can migrate to HTML5 from HTML4, and XHTML, using the same recipe.

Typical HTML4	Typical HTML5
	<header>
	<nav>
	<section>
	<article>
	<footer>

**The Difference Between <article> <section> and **

There is a confusing (lack of) difference in the HTML5 standard, between <article> <section> and .
In the HTML5 standard, the <section> element is defined as a block of related elements.
The <article> element is defined as a complete, self-contained block of related elements.
The element is defined as a block of children elements.
How to interpret that?
In the example above, we have used <section> as a container for related <articles>.
But, we could have used <article> as a container for articles as well.
Here are some different examples:

<article> in <article>:

```
<article>

<h2>Famous Cities</h2>

<article>
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</article>

<article>
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</article>

<article>
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</article>

</article>
```

** in <article>:**

```
<article>

<h2>Famous Cities</h2>

<div class="city">
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</div>

<div class="city">
```

```
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</div>

<div class="city">
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</div>

</article>
```

** in <section> in <article>:**

```
<article>

<section>
<h2>Famous Cities</h2>

<div class="city">
<h2>London</h2>
<p>London is the capital city of England. It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</div>

<div class="city">
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</div>

<div class="city">
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</div>
</section>

<section>
<h2>Famous Countries</h2>

<div class="country">
<h2>England</h2>
<p>London is the capital city of England. It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
</div>

<div class="country">
```

```
<h2>France</h2>
<p>Paris is the capital and most populous city of France.</p>
</div>

<div class="country">
<h2>Japan</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.</p>
</div>
</section>

</article>
```

Grouping Content

1. **p element**
2. **hr element**
3. **pre element**
4. **blockquote element**
5. **ol element**
6. **li element**
7. **dl element**
8. **dt element**
9. **dd element**
10. **Figure element**
11. **figcaption element**
12. **div element**
13. **main element**

p element

The <p> tag defines a paragraph.

A paragraph is marked up as follows:

```
<p>This is some text in a paragraph.</p>
```

hr element

The <hr> tag defines a thematic break in an HTML page (e.g. a shift of topic).The <hr> element is used to separate content (or define a change) in an HTML page.Use the <hr> tag to define a thematic change in the content:

```
<h1>HTML</h1>
<p>HTML is a language for describing web pages.....</p>

<hr>

<h1>CSS</h1>
<p>CSS defines how to display HTML elements.....</p>
```

pre element

The <pre> tag defines preformatted text.

Text in a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.

Preformatted text:

```
<pre>
        Text in a pre element
        is displayed in a fixed-width
        font, and it preserves
        both      spaces and
        line breaks
        </pre>
```

blockquote element

The <blockquote> tag specifies a section that is quoted from another source.Browsers usually indent <blockquote> elements.

A section that is quoted from another source:

```
<blockquote cite="http://www.worldwildlife.org/who/index.html">
        For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization,
        WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5
        million globally.
        </blockquote>
```

Ol element

The <ol> tag defines an ordered list. An ordered list can be numerical or alphabetical.
Use the <li> tag to define list items.

2 different ordered lists:

```
<ol>
        <li>Coffee</li>
        <li>Tea</li>
        <li>Milk</li>
        </ol>

        <ol start="50">
        <li>Coffee</li>
        <li>Tea</li>
        <li>Milk</li>
        </ol>
```

dl element

The <dl> tag defines a description list.
The <dl> tag is used in conjunction with <dt> (defines terms/names) and <dd> (describes each term/name).

A description list, with terms and descriptions:

```
<dl>
        <dt>Coffee</dt>
        <dd>Black hot drink</dd>
        <dt>Milk</dt>
        <dd>White cold drink</dd>
        </dl>
```

dd element

The <dd> tag is used to describe a term/name in a description list.
The <dd> tag is used in conjunction with <dl> (defines a description list) and <dt> (defines terms/names).
Inside a <dd> tag you can put paragraphs, line breaks, images, links, lists, etc.

A description list, with terms and descriptions:

```
<dl>
  <dt>Coffee</dt>
  <dd>Black hot drink</dd>
  <dt>Milk</dt>
  <dd>White cold drink</dd>
</dl>
```

figure element

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.
Use a <figure> element to mark up a photo in a document:

```
<figure>
  <img src="img_pulpit.jpg" alt="The Pulpit Rock" width="304" height="228">
</figure>
```

Embedded Content

14. **img Element**
15. **iframe element**
16. **embed element**
17. **object element**
18. **audio element**
19. **video element**
20. **track element**
21. **map element**
22. **area element**

img Element

- The <img> tag defines an image in an HTML page.
- The <img> tag has two required attributes: src and alt.

Example:

- `<img src="smiley.gif" alt="Smiley face" height="42" width="42">`

iframe Element

The <iframe> tag specifies an inline frame.
An inline frame is used to embed another document within the current HTML document.

embed element

The <embed> tag defines a container for an external application or interactive content (a plug-in).
An embedded flash animation:

```
<embed src="helloworld.swf">
```

object element

The <object> tag defines an embedded object within an HTML document. Use this element to embed multimedia (like audio, video, Java applets, ActiveX, PDF, and Flash) in your web pages. We can also use the <object> tag to embed another webpage into your HTML document. We can use the <param> tag to pass parameters to plugins that have been embedded with the <object> tag.

Example

```
<object width="400" height="400" data="helloworld.swf"></object>
```

audio element

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

Example

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

video element

The <video> tag specifies video, such as a movie clip or other video streams.

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

Play a video:

```
<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>
```

track element

The <track> tag specifies text tracks for media elements (<audio> and <video>).This element is used to specify subtitles, caption files or other files containing text, that should be visible when the media is playing.

A video with two subtitle tracks:

```
<video width="320" height="240" controls>
  <source src="forrest_gump.mp4" type="video/mp4">
  <source src="forrest_gump.ogg" type="video/ogg">
  <track src="subtitles_en.vtt" kind="subtitles" srclang="en" label="English">
  <track src="subtitles_no.vtt" kind="subtitles" srclang="no" label="Norwegian">
</video>
```

map element

The <map> tag is used to define a client-side image-map. An image-map is an image with clickable areas.The required name attribute of the <map> element is associated with the <img>'s usemap attribute and creates a relationship between the image and the map.The <map> element contains a number of <area> elements, that defines the clickable areas in the image map.

An image-map, with clickable areas:

```
<img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap">

<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun">
  <area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury">
  <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus">
</map>
```

area element

The <area> tag defines an area inside an image-map (an image-map is an image with clickable areas).The <area> element is always nested inside a <map> tag.

Note: The usemap attribute in the <img> tag is associated with the <map> element's name attribute, and creates a relationship between the image and the map.

An image-map, with clickable areas:

```
<img src="planets.gif" width="145" height="126" alt="Planets"
usemap="#planetmap">

<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun">
  <area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury">
  <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus">
</map>
```

HTML Graphics

HTML5 Canvas

Canvas is a medium for oil painting. One of the earliest oils on canvas is a French Madonna from 1410. Canvas is typically stretched across a wooden frame.

On the HTML canvas, you can draw all kinds of graphics, from simple lines, to complex graphic objects.

The HTML <canvas> Element

The HTML **<canvas>** element (introduced in HTML5) is a **container** for canvas graphics.
An HTML canvas is a rectangular area on an HTML page.
Canvas has several methods for drawing paths, boxes, circles, text, and graphic images.

Canvas Examples

Basic Canvas Example

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

Drawing with JavaScript

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

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);
```

Stroke Text

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

Draw 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);
```

Draw Circular 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);
```

Draw Image

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

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

The HTML <svg> element (introduced in HTML5) 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

Sorry, your browser does not support inline SVG.

```
<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>
```

SVG Star

Sorry, your browser does not support inline SVG.

```
<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
```

SVG Logo

SVG Sorry, your browser does not support inline SVG.

```
<svg height="130" width="500">
```

```
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana" x="50" y="86">SVG</text>
  Sorry, your browser does not support inline SVG.
</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

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

HTML media

Multimedia on the web, is sound, music, videos, movies, and animations.

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.
Web pages often contains multimedia elements of different types and formats.
In this chapter you will learn about the different multimedia formats.

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.

HTML5 Audio

HTML5 provides a standard for playing audio files.

Audio on the Web

Before HTML5, there was no standard for playing audio files on a web page.
Before HTML5, audio files could only be played with a plug-in (like flash).
The HTML5 <audio> element specifies a standard way to embed audio in a web page.

The HTML <audio> Element

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

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>
```

HTML Audio - How It Works

The **controls** attribute adds audio controls, like play, pause, and volume.
Text between the <audio> and </audio> tags will display in browsers that do not support the <audio> element.
Multiple **<source>** elements can link to different audio files. The browser will use the first recognized format.

HTML Audio - Browser Support

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

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

HTML Audio - Media Types

File Format	Media Type
MP3	audio/mpeg
Ogg	audio/ogg
Wav	audio/wav

HTML Audio - Methods, Properties, and Events

HTML5 defines DOM methods, properties, and events for the <audio> element.
This allows you to load, play, and pause audios, as well as setting duration and volume.
There are also DOM events that can notify you when an audio begins to play, is paused, etc.

HTML5 Audio Tags

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

HTML YouTube Videos

The easiest way to play videos in HTML, is to use YouTube.

Struggling with Video Formats?

Different versions of different browsers support different video formats.
Earlier in this tutorial, you have seen that you might have to convert your videos to different video formats, to make sure they play in all browsers.
Converting videos to different format can be difficult and time consuming.
An easier solution might be to let YouTube play the videos in your web pages.

Frame

The <frame> tag is not supported in HTML5.
The <frame> tag defines one particular window (frame) within a <frameset>.
Each <frame> in a <frameset> can have different attributes, such as border, scrolling, the ability to resize, etc. The <frameset> tag is not supported in HTML5.
The <frameset> tag defines a frameset.
The <frameset> element holds one or more <frame> elements. Each <frame> element can hold a separate document.

The <frameset> element specifies HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them.

```
<frameset cols="25%,50%,25%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">
</frameset>
```

Optional Attributes

Attribute	Value	Description
cols	*pixels* *%* *	Not supported in HTML5. Specifies the number and size of columns in a frameset
rows	*pixels* *%* *	Not supported in HTML5. Specifies the number and size of rows in a frameset

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, in your web page, to display the video:

Example - Using iFrame

```
<iframe width="420" height="315"
src="http://www.youtube.com/embed/XGSy3_Czz8k">
</iframe>
```

Example - Using <object>

```
<object width="420" height="315"
data="http://www.youtube.com/v/XGSy3_Czz8k">
</object>
```

Example - Using <embed>

```
<embed width="420" height="315"
src="http://www.youtube.com/v/XGSy3_Czz8k">
```

1. Interactive Elements
    1. The details element
    2. The summary element
    3. The command element
    4. The menu element

<details> element

The <details> tag specifies additional details that the user can view or hide on demand. The <details> tag can be used to create an interactive widget that the user can open and close. Any sort of content can be put inside the <details> tag.The content of a <details> element should not be visible unless the open attribute is set

Example

```
<details>
  <summary>Copyright 1999-2014.</summary>
  <p> - by Refsnes Data. All Rights Reserved.</p>
  <p>All content and graphics on this web site are the property of the company Refsnes Data.</p>
</details>
```

<summary> element

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

Example as **above**

<menu> element

The <menu> tag defines a list/menu of commands.

The <menu> tag is used for context menus, toolbars and for listing form controls and commands.

<command> element

Command tag is new tag in HTML5 used for performing different action.it is placed inside the menu tag as below example:

Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function save()
{
alert("Some javaScript....");
}
</script>
</head>
<body>

<menu>
<command type="command" label="Save" onclick="save()">Save</command>
```

```
</menu>
</body>
</html>
```