HTML Tutorial

CONTENTS PAGE

XHTML HOME

CSS HOME

CSS Advanced

JS HOME

JS Objects

JS Advanced

Introduction to HTML

What is an HTML File?

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- An HTML file is a text file containing small **markup tags**
- The markup tags tell the Web browser **how to display** the page
- An HTML file must have an **htm** or **html** file extension
- An HTML file can be created using a **simple text editor**

Do You Want to Try It?

If you are running Windows, start Notepad.

If you are on a Mac, start SimpleText.

In OSX start TextEdit and change the following preferences: Open the "Format" menu and select "Plain text" instead of "Rich text". Then open the "Preferences" window under the "Text Edit" menu and select "Ignore rich text commands in HTML files". Your HTML code will probably not work if you do not change the preferences above!

Type in the following text:

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage. <b>This text is bold</b>
</body>
</html>
```

Save the file as "mypage.htm".

Start your Internet browser. Select "Open" (or "Open Page") in the File menu of your browser. A dialog box will appear. Select "Browse" (or "Choose File") and locate the HTML file you just created - "mypage.htm" - select it and click "Open". Now you should see an address in the dialog box, for example "C:\MyDocuments\mypage.htm". Click OK, and the browser will display the page.

Example Explained

The first tag in your HTML document is <html>. This tag tells your browser that this is the start of an HTML document. The last tag in your document is </html>. This tag tells your browser that this is the end of the HTML document.

The text between the <head> tag and the </head> tag is header information. Header information is not displayed in the browser window.

The text between the <title> tags is the title of your document. The title is displayed in your browser's caption.

The text between the <body> tags is the text that will be displayed in your browser.

The text between the and tags will be displayed in a bold font.

HTM or HTML Extension?

When you save an HTML file, you can use either the .htm or the .html extension. We have used .htm in our examples. It might be a bad habit inherited from the past when some of the commonly used software only allowed three letter extensions.

With newer software we think it will be perfectly safe to use .html.

Note on HTML Editors:

You can easily edit HTML files using a WYSIWYG (what you see is what you get) editor like FrontPage or Dreamweaver, instead of writing your markup tags in a plain text file.

However, if you want to be a skillful Web developer, we strongly recommend that you use a plain text editor to learn your primer HTML.

Frequently Asked Questions

Q: After I have edited an HTML file, I cannot view the result in my browser. Why?
A: Make sure that you have saved the file with a proper name and extension like "c:\mypage.htm". Also make sure that you use the same name when you open the file in your browser.

Q: I have edited an HTML file, but the changes don't show in the browser. Why?
A: A browser caches pages so it doesn't have to read the same page twice. When you have modified a page, the browser doesn't know that. Use the browser's refresh/reload button to force the browser to reload the page.

Q: What browser should I use?
A: You can do all the training with all of the well-known browsers, like Internet Explorer, Firefox, Netscape, or Opera. However, some of the examples in our advanced classes require the latest versions of the browsers.

Q: Does my computer have to run Windows? What about a Mac?
A: You can do all your training on a non-Windows computer like a Mac.

HTML Elements

HTML documents are text files made up of HTML elements.

HTML elements are defined using HTML tags.

HTML Tags

- HTML tags are used to mark-up HTML **elements**
- HTML tags are surrounded by the **two characters < and >**
- The surrounding characters are called **angle brackets**
- HTML tags normally **come in pairs** like and
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- The text between the start and end tags is the **element content**
- HTML tags are **not case sensitive,** means the same as <B>

HTML Elements

Remember the HTML example from the previous page:

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage. <b>This text is bold</b>
</body>
</html>
```

This is an HTML element:

```
<b>This text is bold</b>
```

The HTML element starts with a **start tag**:
The **content** of the HTML element is: This text is bold
The HTML element ends with an **end tag**:

The purpose of the tag is to define an HTML element that should be displayed as bold.

This is also an HTML element:

```
<body>
This is my first homepage. <b>This text is bold</b>
</body>
```

This HTML element starts with the start tag <body>, and ends with the end tag </body>.

The purpose of the <body> tag is to define the HTML element that contains the body of the HTML document.

Why do We Use Lowercase Tags?

We have just said that HTML tags are not case sensitive: <B> means the same as <b>. If you surf the Web, you will notice that plenty of web sites use uppercase HTML tags in their source code. We always use lowercase tags. Why?

If you want to follow the latest web standards, you should always use lowercase tags. The World Wide Web Consortium (W3C) recommends lowercase tags in their HTML 4 recommendation, and XHTML (the next generation HTML) demands lowercase tags.

Basic HTML Tags

The most important tags in HTML are tags that define headings, paragraphs and line breaks.

The best way to learn HTML is to work with examples. We have created a very nice HTML editor for you. With this editor, you can edit the HTML source code if you like, and click on a test button to view the result.

Try it Yourself - Examples

A very simple HTML document
This example is a very simple HTML document, with only a minimum of HTML tags. It demonstrates how the text inside a body element is displayed in the browser.

<html><body>The content of the body element is displayed in your browser.</body></html>

Simple paragraphs
This example demonstrates how the text inside paragraph elements is displayed in the browser.

<html><body> This is a paragraph. This is a paragraph.

This is a paragraph. Paragraph elements are defined by the p tag.

</body></html>

Headings

Headings are defined with the <h1> to <h6> tags. <h1> defines the largest heading. <h6> defines the smallest heading.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
```

```
<h3>This is a heading</h3>
<h4>This is a heading</h4>
<h5>This is a heading</h5>
<h6>This is a heading</h6>
```

HTML automatically adds an extra blank line before and after a heading.

Paragraphs

Paragraphs are defined with the tag.

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

HTML automatically adds an extra blank line before and after a paragraph.

Don't Forget the Closing Tag

You might have noticed that paragraphs can be written without end tags :

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

The example above will work in most browsers, but don't rely on it. Future version of HTML will not allow you to skip ANY end tags.

Closing all HTML elements with an end tag is a future-proof way of writing HTML. It also makes the code easier to understand (read and browse) when you mark both where an element starts and where it ends.

Line Breaks

The
 tag is used when you want to break a line, but don't want to start a new paragraph. The
 tag forces a line break wherever you place it.

```
<p>This <br> is a para<br>graph with line breaks</p>
```

The
 tag is an empty tag. It has no end tag like </br>, since a closing tag doesn't make any sense.

**
 or
**

More and more often you will see the
 tag written like this:

Because the
 tag has no end tag (or closing tag), it breaks one of the rules for future HTML (the XML based XHTML), namely that all elements must be closed.

Writing it like
 is a future proof way of closing (or ending) the tag inside the opening tag, accepted by both HTML and XML.

Comments in HTML

The comment tag is used to insert a comment in the HTML source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.

```
<!-- This is a comment -->
```

Note that you need an exclamation point after the opening bracket, but not before the closing bracket.

Recap on HTML Elements

- Each HTML element has **an element name** (body, h1, p, br)
- The **start tag is the name** surrounded by angle brackets: <h1>
- The **end tag is a slash and the name** surrounded by angle brackets </h1>
- **The element content** occurs between the start tag and the end tag
- Some HTML elements have no content
- Some HTML elements have no end tag

Basic Notes - Useful Tips

When you write HTML text, you can never be sure how the text is displayed in another browser. Some people have large computer displays, some have small. The text will be reformatted every time the user resizes his window. Never try to format the text in your editor by adding empty lines and spaces to the text.

HTML will truncate the spaces in your text. Any number of spaces count as one. Some extra information: In HTML a new line counts as one space.

Using empty paragraphs <p> to insert blank lines is a bad habit. Use the
 tag instead. (But don't use the
 tag to create lists. Wait until you have learned about HTML lists.)

HTML automatically adds an extra blank line before and after some elements, like before and after a paragraph, and before and after a heading.

We use a horizontal rule (the <hr> tag), to separate the sections in our tutorials.

More Examples

More paragraphs
This example demonstrates some of the default behaviors of paragraph elements.

```
<html><body>

<p>This paragraph contains a lot of lines in the source code, but the browser  ignores it. </p>

<p>This paragraph contains a lot of spaces in the source code, but the browser ignores it.</p>

<p>The number of lines in a paragraph depends on the size of your browser window. If you resize the browser window, the number of lines in this paragraph will change. </p>

</body></html>
```

Line breaks
This example demonstrates the use of line breaks in an HTML document.

```
<html><body>

<p>To break<br>lines<br>in a<br>paragraph,<br>use the br tag.</p>

</body></html>
```

Poem problems
This example demonstrates some problems with HTML formatting.

```
<html><body>

<p>My Bonnie lies over the ocean. My Bonnie lies over the sea. My Bonnie lies over the ocean.

  Oh, bring back my Bonnie to me.</p>

<p>Note that your browser simply ignores your formatting!</p>

</body></html>
```

Headings
This example demonstrates the tags that display headings in an HTML document.

```
<html><body><h1>This is heading 1</h1><h2>This is heading 2</h2>

<h3>This is heading 3</h3><h4>This is heading 4</h4><h5>This is heading 5</h5>

<h6>This is heading 6</h6>

<p>Use heading tags only for headings. Don't use them just to make something bold. Use other tags for that.</p></body></html>
```

Horizontal rule
This example demonstrates how to insert a horizontal rule.

```
<html><body><p>The hr tag defines a horizontal rule:</p><hr><p>This is a paragraph</p><hr>

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

Hidden comments
This example demonstrates how to insert a hidden comment in the HTML source code.

<html><body><!--This comment will not be displayed-->

This is a regular paragraph </body></html>

Basic HTML Tags

If you lookup the basic HTML tags in the reference below, you will see that the reference contains additional information about tag attributes.

You will learn more about HTML tag attributes in the next chapter of this tutorial.

Tag	Description
<html>	Defines an HTML document
<body>	Defines the document's body
<h1> to <h6>	Defines header 1 to header 6
	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a horizontal rule
<!-->	Defines a comment

HTML Attributes

Attributes provide additional information to an HTML element.

HTML Tag Attributes

HTML tags can have attributes. Attributes provide additional information to an HTML element.

Attributes always come in name/value pairs like this: name="value".

Attributes are always specified in the start tag of an HTML element.

Attributes Example 1:

<h1> defines the start of a heading.

<h1 align="center"> has additional information about the alignment.

<html><body><h1 align="center">This is heading 1</h1>

The heading above is aligned to the center of this page. The heading above is aligned to the center of this page. The heading above is aligned to the center of this page.

</body></html>

Attributes Example 2:

<body> defines the body of an HTML document.

<body bgcolor="yellow"> has additional information about the background color.

<html><body bgcolor="yellow"><h2>Look: Colored Background!</h2></body></html>

Attributes Example 3:

<table> defines an HTML table. (You will learn more about HTML tables later)

<table border="1"> has additional information about the border around the table.

Use Lowercase Attributes

Attributes and attribute values are case-insensitive. However, the World Wide Web Consortium (W3C) recommends lowercase attributes/attribute values in their HTML 4 recommendation, and XHTML demands lowercase attributes/attribute values.

Always Quote Attribute Values

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed.

In some rare situations, like when the attribute value itself contains quotes, it is necessary to use single quotes:

name='John "ShotGun" Nelson'

HTML Text Formatting

HTML defines a lot of elements for formatting output, like bold or italic text.

Below are a lot of examples that you can try out yourself:

Examples

Text formatting
This example demonstrates how you can format text in an HTML document.

```
<html><body><b>This text is bold</b><br><strong>This text is strong</strong><br>

<big>This text is big</big><br><em>This text is emphasized</em><br>

<i>This text is italic</i><br>

<small>This text is small</small><br>

This text contains <sub>subscript</sub><br>This text contains<sup>superscript</sup>

</body></html>
```

Preformatted text
This example demonstrates how you can control the line breaks and spaces with the pre tag.

```
<html><body><pre>This is preformatted text. It preserves both spaces and line breaks.</pre>

<p>The pre tag is good for displaying computer code:</p><pre>

for i = 1 to 10     print i  next i  </pre> </body> </html>
```

"Computer output" tags
This example demonstrates how different "computer output" tags will be displayed.

```
<html><body><code>Computer code</code><br><kbd>Keyboard input</kbd><br>

<tt>Teletype text</tt><br><samp>Sample text</samp><br><var>Computer variable</var>

<br><p><b>Note:</b> These tags are often used to display computer/programming code.</p>

</body></html>
```

Address
This example demonstrates how to write an address in an HTML document.

```
<html><body><address>Donald Duck<br>BOX 555<br>Disneyland<br>USA</address>

</body></html>
```

Abbreviations and acronyms
This example demonstrates how to handle an abbreviation or an acronym.

```
<html><body><abbr title="United Nations">UN</abbr><br>

<acronym title="World Wide Web">WWW</acronym>

<p>The title attribute is used to show the spelled-out version when holding the mouse pointer over the acronym or abbreviation.</p><p>This only works for the acronym element in IE 5.</p>

<p>This works for both the abbr and acronym element in Netscape 6.2.</p></body></html>
```

Text direction
This example demonstrates how to change the text direction.

<html><body>

If your browser supports bi-directional override (bdo), the next line will be written from the right to the left (rtl):

<bdo dir="rtl">Here is some Hebrew text</bdo></body></html>

Quotations
This example demonstrates how to handle long and short quotations.

<html><body>Here comes a long quotation:<blockquote>This is a long quotation. This is a long quotation. This is a long quotation. This is a long quotation. This is a long quotation.</blockquote>

Here comes a short quotation:<q>This is a short quotation</q>

With the block quote element, the browser inserts line breaks and margins, but the q element does not render as anything special. </body></html>

Deleted and inserted text
This example demonstrates how to mark a text that is deleted or inserted to a document.

<html><body> a dozen is ~~twenty~~ <ins>twelve</ins> pieces

Most browsers will overstrike deleted text and underline inserted text.

Some older browsers will display deleted or inserted text as plain text. </body></html>

How to View HTML Source

Have you ever seen a Web page and wondered "Hey! How did they do that?"

To find out, click the VIEW option in your browser's toolbar and select SOURCE or PAGE SOURCE. This will open a window that shows you the HTML code of the page.

Text Formatting Tags

Tag	Description
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<s>	Deprecated. Use <del> instead

<strike>	Deprecated. Use <del> instead
<u>	Deprecated. Use styles instead

"Computer Output" Tags

Tag	Description
<code>	Defines computer code text
<kbd>	Defines keyboard text
<samp>	Defines sample computer code
<tt>	Defines teletype text
<var>	Defines a variable
<pre>	Defines preformatted text
<listing>	Deprecated. Use <pre> instead
<plaintext>	Deprecated. Use <pre> instead
<xmp>	Deprecated. Use <pre> instead

Citations, Quotations, and Definition Tags

Tag	Description
<abbr>	Defines an abbreviation
<acronym>	Defines an acronym
<address>	Defines an address element
<bdo>	Defines the text direction
<blockquote>	Defines a long quotation
<q>	Defines a short quotation
<cite>	Defines a citation
<dfn>	Defines a definition term

HTML Character Entities

Some characters like the < character, have a special meaning in HTML, and therefore cannot be used in the text.

To display a less than sign (<) in HTML, we have to use a character entity.

Character Entities

Some characters have a special meaning in HTML, like the less than sign (<) that defines the start of an HTML tag. If we want the browser to actually display these characters we must insert character entities in the HTML source.

A character entity has three parts: an ampersand (&), an entity name or a # and an entity number, and finally a semicolon (;).

To display a less than sign in an HTML document we must write: **<** or **<**

The advantage of using a name instead of a number is that a name is easier to remember. The disadvantage is that not all browsers support the newest entity names, while the support for entity numbers is very good in almost all browsers.

Note that the entities are case sensitive.

This example lets you experiment with character entities: Character Entities

<html><body> Character entities &X; Substitute the "X" with an entity number like "#174" or an entity name like "pound" to see the result. </body></html>

Non-breaking Space

The most common character entity in HTML is the non-breaking space.

Normally HTML will truncate spaces in your text. If you write 10 spaces in your text HTML will remove 9 of them. To add spaces to your text, use the   character entity.

The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
"	quotation mark	"	"
'	apostrophe	' (does not work in IE)	'

Some Other Commonly Used Character Entities:

Result	Description	Entity Name	Entity Number
¢	cent	¢	¢
£	pound	£	£
¥	yen	¥	¥
€	euro	€	€
§	section	§	§
©	copyright	©	©
®	registered trademark	®	®
×	multiplication	×	×
÷	division	÷	÷

To see a full list of HTML character entities go to our HTML Entities Reference.

HTML Links

HTML uses a hyperlink to link to another document on the Web.

Examples

Create hyperlinks
This example demonstrates how to create links in an HTML document.

<html><body> This text is a link to a page on this Web site.

This text is a link to a page on the World Wide Web.

</body></html>

An image as a link
This example demonstrates how to use an image as a link.

<html><body> You can also use an image as a link:

<img border="0" src="buttonnext.gif" width="65" height="38"> </body></html>

The Anchor Tag and the Href Attribute

HTML uses the <a> (anchor) tag to create a link to another document.

An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc.

The syntax of creating an anchor:

```
<a href="url">Text to be displayed</a>
```

The <a> tag is used to create an anchor to link from, the href attribute is used to address the document to link to, and the words between the open and close of the anchor tag will be displayed as a hyperlink.

This anchor defines a link to W3Schools:

```
<a href="http://www.w3schools.com/">Visit W3Schools!</a>
```

The line above will look like this in a browser:

The Target Attribute

With the target attribute, you can define **where** the linked document will be opened.

The line below will open the document in a new browser window:

```
<a href="http://www.w3schools.com/"
target="_blank">Visit W3Schools!</a>
```

The Anchor Tag and the Name Attribute

The name attribute is used to create a named anchor. When using named anchors we can create links that can jump directly into a specific section on a page, instead of letting the user scroll around to find what he/she is looking for.

Below is the syntax of a named anchor:

```
<a name="label">Text to be displayed</a>
```

The name attribute is used to create a named anchor. The name of the anchor can be any text you care to use.

The line below defines a named anchor:

```
<a name="tips">Useful Tips Section</a>
```

You should notice that a named anchor is not displayed in a special way.

To link directly to the "tips" section, add a # sign and the name of the anchor to the end of a URL, like this:

```
<a href="http://www.w3schools.com/html_links.asp#tips">
Jump to the Useful Tips Section</a>
```

A hyperlink to the Useful Tips Section from WITHIN the file "html_links.asp" will look like this:

```
<a href="#tips">Jump to the Useful Tips Section</a>
```

Basic Notes - Useful Tips

Always add a trailing slash to subfolder references. If you link like this:
href="http://www.w3schools.com/html", you will generate two HTTP requests to the server, because the server will add a slash to the address and create a new request like this:
href="http://www.w3schools.com/html/"

Named anchors are often used to create "table of contents" at the beginning of a large document. Each chapter within the document is given a named anchor, and links to each of these anchors are put at the top of the document.

If a browser cannot find a named anchor that has been specified, it goes to the top of the document. No error occurs.

More Examples

Open a link in a new browser window
This example demonstrates how to link to another page by opening a new window, so that the visitor does not have to leave your Web site.

```
<html><body><a href="lastpage.htm" target="_blank">Last Page</a> <p>If you set the target attribute of a link to "_blank",the link will open in a new window.</p></body></html>
```

Link to a location on the same page
This example demonstrates how to use a link to jump to another part of a document.

```
<html><body><p><a href="#C4">See also Chapter 4.</a></p><h2>Chapter 1</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 2</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 3</h2>

<p>This chapter explains ba bla bla</p><h2><a name="C4">Chapter 4</a></h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 5</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 6</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 7</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 8</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 9</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 10</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 11</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 12</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 13</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 14</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 15</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 16</h2>

<p>This chapter explains ba bla bla</p><h2>Chapter 17</h2>

<p>This chapter explains ba bla bla</p></body></html>
```

Break out of a frame
This example demonstrates how to break out of a frame, if your site is locked in a frame.

```
<html><body><p>Locked in a frame?</p> <a href="http://www.w3schools.com/"
```

target="_top">Click here! </body></html>

Create a mailto link
This example demonstrates how to link to a mail message (will only work if you have mail installed).

<html><body> This is a mail link:

Send Mail

Note: Spaces between words should be replaced by %20 to ensure that the browser will display your text properly. </body></html>

Create a mailto link 2
This example demonstrates a more complicated mailto link.

<html><body> This is another mailto link:

<a
href="mailto:someone@microsoft.com?cc=someoneelse@microsoft.com&bcc=andsomeoneelse2@microsoft.com&subject=Summer%20Party&body=You%20are%20invited%20to%20a%20big%20summer%20party!">Send mail!

Note: Spaces between words should be replaced by %20 to ensure that the browser will display your text properly. </body></html>

Link Tags

Tag	Description
<a>	Defines an anchor

HTML Frames

With frames, you can display more than one Web page in the same browser window.

Examples

Vertical frameset
This example demonstrates how to make a vertical frameset with three different documents.

<html><frameset cols="25%,50%,25%"> <frame src="frame_a.htm">

<frame src="frame_b.htm"> <frame src="frame_c.htm"></frameset></html>

Horizontal frameset
This example demonstrates how to make a horizontal frameset with three different documents.

<html><frameset rows="25%,50%,25%"> <frame src="frame_a.htm">

<frame src="frame_b.htm"> <frame src="frame_c.htm"></frameset></html>

Frames

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

- The web developer must keep track of more HTML documents
- It is difficult to print the entire page

The Frameset Tag

- The <frameset> tag defines how to divide the window into frames
- Each frameset defines a set of rows **or** columns
- The values of the rows/columns indicate the amount of screen area each row/column will occupy

---

The Frame Tag

- The <frame> tag defines what HTML document to put into each frame

In the example below we have a frameset with two columns. The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The HTML document "frame_a.htm" is put into the first column, and the HTML document "frame_b.htm" is put into the second column:

```
<frameset cols="25%,75%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
</frameset>
```

Note: The frameset column size value can also be set in pixels (cols="200,500"), and one of the columns can be set to use the remaining space (cols="25%,*").

Basic Notes - Useful Tips

If a frame has visible borders, the user can resize it by dragging the border. To prevent a user from doing this, you can add noresize="noresize" to the <frame> tag.

Add the <noframes> tag for browsers that do not support frames.

Important: You cannot use the <body></body> tags together with the <frameset></frameset> tags! However, if you add a <noframes> tag containing some text for browsers that do not support frames, you will have to enclose the text in <body></body> tags! See how it is done in the first example below.

More Examples

How to use the <noframes> tag
This example demonstrates how to use the <noframes> tag.

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

  <frame src="frame_b.htm">  <frame src="frame_c.htm"><noframes>

<body>Your browser does not handle frames!</body></noframes></frameset></html>
```

Mixed frameset
This example demonstrates how to make a frameset with three documents, and how to mix them in rows and columns.

```
<html><frameset rows="50%,50%"><frame src="frame_a.htm"><frameset cols="25%,75%">

<frame src="frame_b.htm"><frame src="frame_c.htm"></frameset></frameset></html>
```

Frameset with noresize="noresize"
This example demonstrates the noresize attribute. The frames are not resizable. Move the mouse over the borders between the frames and notice that you can not move the borders.

```
<html><frameset rows="50%,50%"><frame noresize="noresize" src="frame_a.htm">

<frameset cols="25%,75%"><frame noresize="noresize" src="frame_b.htm">

<frame noresize="noresize" src="frame_c.htm"></frameset></frameset></html>
```

Navigation frame
This example demonstrates how to make a navigation frame. The navigation frame contains a list of links with the second frame as the target. The file called "tryhtml_contents.htm" contains three links. The source code of the links:

```
<a href ="frame_a.htm" target ="showframe">Frame a</a><br>
<a href ="frame_b.htm" target ="showframe">Frame b</a><br>
<a href ="frame_c.htm" target ="showframe">Frame c</a>
```

The second frame will show the linked document.

```
<html><frameset cols="120,*"><frame src="tryhtml_contents.htm"><frame src="frame_a.htm"

name="showframe"></frameset></html>
```

Inline frame
This example demonstrates how to create an inline frame (a frame inside an HTML page).

```
<html><body><iframe src="default.asp"></iframe>

<p>Some older browsers don't support iframes.</p>

<p>If they don't, the iframe will not be visible.</p></body></html>
```

Jump to a specified section within a frame
This example demonstrates two frames. One of the frames has a source to a specified section in a file. The specified section is made with <a name="C10"> in the "link.htm" file.

```
<html><frameset cols="20%,80%"> <frame src="frame_a.htm"> <frame src="link.htm#C10">

</frameset></html>
```

Jump to a specified section with frame navigation
This example demonstrates two frames. The navigation frame (content.htm) to the left contains a list of links with the second frame (link.htm) as a target. The second frame shows the linked document. One of the links in the navigation frame is linked to a specified section in the target file. The HTML code in the file "content.htm" looks like this: Link without Anchor
Link with Anchor.

```
<html><frameset cols="180,*"><frame src="content.htm">

<frame src="link.htm" name="showframe"></frameset></html>
```

Frame Tags

Tag	Description
<frameset>	Defines a set of frames
<frame>	Defines a sub window (a frame)
<noframes>	Defines a noframe section for browsers that do not handle frames
<iframe>	Defines an inline sub window (frame)

HTML Tables

With HTML you can create tables.

Examples

Tables
This example demonstrates how to create tables in an HTML document.

```
<html><body><p>Each table starts with a table tag. Each table row starts with a tr tag.

Each table data starts with a td tag.</p><h4>One column:</h4><table border="1"><tr>

<td>100</td></tr></table><h4>One row and three columns:</h4><table border="1">

<tr><td>100</td><td>200</td><td>300</td></tr></table><h4>Two rows and three columns:</h4><table border="1"><tr>  <td>100</td>  <td>200</td><td>300</td></tr><tr>
<td>400</td>  <td>500</td>  <td>600</td></tr></table></body></html>
```

Table borders

This example demonstrates different table borders.

```
<html><body><h4>With a normal border:</h4>  <table border="1"><tr>  <td>First</td>

<td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table>

<h4>With a thick border:</h4><table border="8"><tr><td>First</td><td>Row</td></tr><tr>

<td>Second</td><td>Row</td></tr></table><h4>With a very thick border:</h4>

<table border="15"><tr><td>First</td><td>Row</td></tr><tr><td>Second</td>

<td>Row</td></tr></table></body></html>
```

Tables

Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). The letters td stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Tables and the Border Attribute

If you do not specify a border attribute the table will be displayed without any borders. Sometimes this can be useful, but most of the time, you want the borders to show.

To display a table with borders, you will have to use the border attribute:

```
<table border="1">
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
```

```
</tr>
</table>
```

Headings in a Table

Headings in a table are defined with the <th> tag.

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Empty Cells in a Table

Table cells with no content are not displayed very well in most browsers.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td></td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Note that the borders around the empty table cell are missing (NB! Mozilla Firefox displays the border).

To avoid this, add a non-breaking space ( ) to empty data cells, to make the borders visible:

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>&amp;nbsp;</td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Basic Notes - Useful Tips

The <thead>,<tbody> and <tfoot> elements are seldom used, because of bad browser support. Expect this to change in future versions of XHTML. If you have Internet Explorer 5.0 or newer, you can view a working example in our XML tutorial.

More Examples

Table with no border
This example demonstrates a table with no borders.

<html><body><h4>This table has no borders:</h4><table><tr><td>100</td><td>200</td>

<td>300</td></tr><tr><td>400</td><td>500</td><td>600</td></tr></table>

<h4>And this table has no borders:</h4><table border="0"><tr><td>100</td><td>200</td>

<td>300</td></tr><tr><td>400</td><td>500</td><td>600</td></tr></table></body>

</html>

Headings in a table
This example demonstrates how to display table headers.

<html><body><h4>Table headers:</h4><table border="1"><tr><th>Name</th>

<th>Telephone</th><th>Telephone</th></tr><tr><td>Bill Gates</td><td>555 77 854</td>

<td>555 77 855</td></tr></table><h4>Vertical headers:</h4><table border="1"><tr>

<th>First Name:</th><td>Bill Gates</td></tr><tr><th>Telephone:</th><td>555 77 854</td>

</tr><tr><th>Telephone:</th><td>555 77 855</td></tr></table></body></html>

Empty cells
This example demonstrates how to use " " to handle cells that have no content.

<html><body><table border="1"><tr><td>Some text</td><td>Some text</td></tr><tr>
<td></td><td>Some text</td></tr></table>

As you can see, one of the cells has no border. That is because it is empty. Try to insert a space in the cell. Still it has no border.

The trick is to insert a no-breaking space in the cell.

No-breaking space is a character entity. If you don't know what a character entity is, read the chapter about it.

The no-breaking space entity starts with an ampersand ("&"),then the letters "nbsp", and ends with a semicolon (";")

 </body></html>

Table with a caption
This example demonstrates a table with a caption.

<html><body><h4>This table has a caption,and a thick border:</h4><table border="6">

<caption>My Caption</caption><tr><td>100</td><td>200</td><td>300</td></tr><tr>

<td>400</td><td>500</td><td>600</td></tr></table></body></html>

Table cells that span more than one row/column
This example demonstrates how to define table cells that span more than one row or one column.

<html><body><h4>Cell that spans two columns:</h4><table border="1"><tr><th>Name</th>

<th colspan="2">Telephone</th></tr><tr><td>Bill Gates</td><td>555 77 854</td>

<td>555 77 855</td></tr></table><h4>Cell that spans two rows:</h4><table border="1">

<tr><th>First Name:</th><td>Bill Gates</td></tr><tr><th rowspan="2">Telephone:</th>

<td>555 77 854</td></tr><tr><td>555 77 855</td></tr></table></body></html>

Tags inside a table
This example demonstrates how to display elements inside other elements.

<html><body><table border="1"><tr><td> This is a paragraph

This is another paragraph </td><td>This cell contains a table:<table border="1"><tr>

<td>A</td> <td>B</td></tr><tr><td>C</td><td>D</td></tr></table></td></tr><tr>

<td>This cell contains a list
apples
bananas
pineapples</td>

<td>HELLO</td></tr></table></body></html>

Cell padding
This example demonstrates how to use cellpadding to create more white space between the cell content and its borders.

<html><body><h4>Without cellpadding:</h4><table border="1"><tr><td>First</td>

<td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table>

<h4>With cellpadding:</h4><table border="1" cellpadding="10"><tr><td>First</td>

<td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table></body></html>

Cell spacing
This example demonstrates how to use cellspacing to increase the distance between the cells.

<html><body><h4>Without cellspacing:</h4><table border="1"><tr><td>First</td>

<td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table>

<h4>With cellspacing:</h4><table border="1" cellspacing="10"><tr><td>First</td>

<td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table></body></html>

Add a background color or a background image to a table
This example demonstrates how to add a background to a table.

<html><body><h4>A background color:</h4><table border="1" bgcolor="red"><tr>

<td>First</td><td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table>

<h4>A background image:</h4><table border="1" background="bgdesert.jpg"><tr>

<td>First</td><td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table></body>

</html>

Add a background color or a background image to a table cell
This example demonstrates how to add a background to one or more table cells.

<html><body><h4>Cell backgrounds:</h4><table border="1"><tr><td bgcolor="red">First</td>

<td>Row</td></tr><tr><td background="bgdesert.jpg">Second</td><td>Row</td></tr>

</table></body></html>

Align the content in a table cell

This example demonstrates how to use the "align" attribute to align the content of cells, to create a "nice-looking" table.

```
<html><body><table width="400" border="1"> <tr><th align="left">Money spent on....</th>

<th align="right">January</th><th align="right">February</th></tr><tr>

<td align="left">Clothes</td><td align="right">$241.10</td><td align="right">$50.20</td></tr>

<tr><td align="left">Make-Up</td><td align="right">$30.00</td><td align="right">$44.45</td>

</tr><tr><td align="left">Food</td><td align="right">$730.40</td>

<td align="right">$650.00</td></tr><tr><th align="left">Sum</th>

<th align="right">$1001.50</th><th align="right">$744.65</th></tr></table></body></html>
```

The frame attribute

This example demonstrates how to use the "frame" attribute to control the borders around the table.

```
<html><body>

<p>If you see no frames around the tables in these examples, your browser is too old, or does not support it.</p>

<h4>With frame="border":</h4><table frame="border"><tr><td>First</td><td>Row</td></tr>

<tr><td>Second</td><td>Row</td></tr></table><h4>With frame="box":</h4>

<table frame="box"><tr><td>First</td><td>Row</td></tr><tr><td>Second</td>
<td>Row</td></tr></table><h4>With frame="void":</h4><table frame="void"><tr>

<td>First</td><td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table>

<h4>With frame="above":</h4><table frame="above"><tr><td>First</td><td>Row</td></tr>

<tr><td>Second</td><td>Row</td></tr></table><h4>With frame="below":</h4>

<table frame="below"><tr><td>First</td><td>Row</td></tr><tr><td>Second</td>

<td>Row</td></tr></table><h4>With frame="hsides":</h4><table frame="hsides"><tr>

<td>First</td><td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table>

<h4>With frame="vsides":</h4><table frame="vsides"><tr><td>First</td><td>Row</td></tr>

<tr><td>Second</td><td>Row</td></tr></table><h4>With frame="lhs":</h4>

<table frame="lhs"><tr><td>First</td><td>Row</td></tr><tr><td>Second</td><td>Row</td>

</tr></table><h4>With frame="rhs":</h4><table frame="rhs"><tr><td>First</td>
```

<td>Row</td></tr><tr><td>Second</td><td>Row</td></tr></table></body></html>

The frame and border attributes
How to use the "frame" and "border" attributes to control the borders around the table.

<html><body>

If you see no frames around the tables in these examples, your browser does not support the frame attribute.

<table frame="hsides" border="3"><tr><td>First row</td></tr></table>

<table frame="vsides" border="3"><tr><td>First row</td></tr></table></body></html>

Table Tags

Tag	Description
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table
<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer

HTML Lists

HTML supports ordered, unordered and definition lists.

Examples

An unordered list
This example demonstrates an unordered list.

<html><body><h4>An Unordered List:</h4>
Coffee
Tea
Milk

</body></html>

An ordered list
This example demonstrates an ordered list.

<html><body><h4>An Ordered List:</h4>
Coffee
Tea
Milk

</body></html>

Unordered Lists

An unordered list is a list of items. The list items are marked with bullets (typically small black circles).

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

Here is how it looks in a browser:

- Coffee
- Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Ordered Lists

An ordered list is also a list of items. The list items are marked with numbers.

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

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

Here is how it looks in a browser:

1. Coffee
2. Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Definition Lists

A definition list is **not** a list of items. This is a list of terms and explanation of the terms.

A definition list starts with the <dl> tag. Each definition-list term starts with the <dt> tag. Each definition-list definition starts with the <dd> tag.

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

Here is how it looks in a browser:

Coffee
    Black hot drink
Milk
    White cold drink

Inside a definition-list definition (the <dd> tag) you can put paragraphs, line breaks, images, links, other lists, etc.

More Examples

Different types of ordered lists
This example demonstrates different types of ordered lists.

```
<html><body><h4>Numbered list:</h4><ol><li>Apples</li><li>Bananas</li><li>Lemons</li>
<li>Oranges</li></ol><h4>Letters list:</h4><ol type="A"><li>Apples</li><li>Bananas</li>

<li>Lemons</li><li>Oranges</li></ol><h4>Lowercase letters list:</h4><ol type="a">

<li>Apples</li><li>Bananas</li><li>Lemons</li><li>Oranges</li></ol>

<h4>Roman numbers list:</h4><ol type="I"><li>Apples</li><li>Bananas</li><li>Lemons</li>

<li>Oranges</li></ol><h4>Lowercase Roman numbers list:</h4><ol type="i"><li>Apples</li>

<li>Bananas</li><li>Lemons</li><li>Oranges</li></ol></body></html>
```

Different types of unordered Lists
This example demonstrates different types of unordered lists.

```
<html><body><h4>Disc bullets list:</h4><ul type="disc"><li>Apples</li><li>Bananas</li>

<li>Lemons</li><li>Oranges</li></ul><h4>Circle bullets list:</h4><ul type="circle">

<li>Apples</li><li>Bananas</li><li>Lemons</li><li>Oranges</li></ul>

<h4>Square bullets list:</h4><ul type="square"><li>Apples</li><li>Bananas</li>

<li>Lemons</li><li>Oranges</li></ul></body></html>
```

Nested list
This example demonstrates how you can nest lists.

```
<html><body><h4>A nested List:</h4><ul><li>Coffee</li><li>Tea<ul><li>Black tea</li>
<li>Green tea</li></ul></li><li>Milk</li></ul></body></html>
```

Nested list 2
This example demonstrates a more complicated nested list.

```
<html><body><h4>A nested List:</h4><ul><li>Coffee</li><li>Tea<ul><li>Black tea</li>
<li>Green tea<ul><li>China</li><li>Africa</li></ul></li></ul></li><li>Milk</li></ul></body>

</html>
```

Definition list
This example demonstrates a definition list.

```
<html><body><h4>A Definition List:</h4><dl><dt>Coffee</dt><dd>Black hot drink</dd>
<dt>Milk</dt><dd>White cold drink</dd></dl></body></html>
```

List Tags

Tag	Description
	Defines an ordered list
	Defines an unordered list
	Defines a list item
<dl>	Defines a definition list
<dt>	Defines a definition term
<dd>	Defines a definition description
<dir>	Deprecated. Use instead
<menu>	Deprecated. Use instead

HTML Forms and Input

HTML Forms are used to select different kinds of user input.

Examples

Text fields
This example demonstrates how to create text fields on an HTML page. A user can write text in a text field.

```
<html><body><form action="">First name:<input type="text" name="firstname"><br>

Last name: <input type="text" name="lastname"></form></body></html>
```

Password fields

This example demonstrates how to create a password field on an HTML page.

```
<html><body><form action="">Username: <input type="text" name="user"><br>

Password: <input type="password" name="password"></form>

<p>Note that when you type characters in a password field, the browser displays asterisks or bullets instead of the characters.</p></body></html>
```

Forms

A form is an area that can contain form elements.

Form elements are elements that allow the user to enter information (like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.) in a form.

A form is defined with the <form> tag.

```
<form>
  <input>
  <input>
</form>
```

Input

The most used form tag is the <input> tag. The type of input is specified with the type attribute. The most commonly used input types are explained below.

Text Fields

Text fields are used when you want the user to type letters, numbers, etc. in a form.

```
<form>
First name:
<input type="text" name="firstname">
<br>
Last name:
<input type="text" name="lastname">
</form>
```

How it looks in a browser:

First name:

Last name:

Note that the form itself is not visible. Also note that in most browsers, the width of the text field is 20 characters by default.

Radio Buttons

Radio Buttons are used when you want the user to select one of a limited number of choices.

```
<form>
<input type="radio" name="sex" value="male"> Male
<br>
<input type="radio" name="sex" value="female"> Female
</form>
```

How it looks in a browser:

Male

Female

Note that only one option can be chosen.

Checkboxes

Checkboxes are used when you want the user to select one or more options of a limited number of choices.

```
<form>
I have a bike:
<input type="checkbox" name="vehicle" value="Bike">
<br>
I have a car:
<input type="checkbox" name="vehicle" value="Car">
<br>
I have an airplane:
<input type="checkbox" name="vehicle" value="Airplane">
</form>
```

How it looks in a browser:

I have a bike:

I have a car:

I have an airplane:

The Form's Action Attribute and the Submit Button

When the user clicks on the "Submit" button, the content of the form is sent to another file. The form's action attribute defines the name of the file to send the content to. The file defined in the action attribute usually does something with the received input.

```
<form name="input" action="html_form_action.asp"
method="get">
Username:
<input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

How it looks in a browser:

Username: Submit

If you type some characters in the text field above, and click the "Submit" button, you will send your input to a page called "html_form_action.asp". That page will show you the received input.

More Examples

Checkboxes
This example demonstrates how to create check-boxes on an HTML page. A user can select or unselect a checkbox.

<html><body><form action="">I have a bike:

<input type="checkbox" name="vehicle" value="Bike">

I have a car: <input type="checkbox" name="vehicle" value="Car">

I have an airplane: <input type="checkbox" name="vehicle" value="Airplane"></form></body>

</html>

Radio buttons
This example demonstrates how to create radio-buttons on an HTML page.

<html><body><form action="">

Male: <input type="radio" checked="checked"name="Sex" value="male">

Female: <input type="radio"name="Sex" value="female"></form>

When a user clicks on a radio-button, the button becomes checked, and all other buttons with the same name become unchecked </body></html>

Simple drop down box
This example demonstrates how to create a simple drop-down box on an HTML page. A drop-down box is a selectable list.

<html><body><form action=""><select name="cars"><option value="volvo">Volvo</option>

<option value="saab">Saab</option><option value="fiat">Fiat</option>

<option value="audi">Audi</option></select></form></body></html>

Another drop down box
This example demonstrates how to create a simple drop-down box with a pre-selected value.

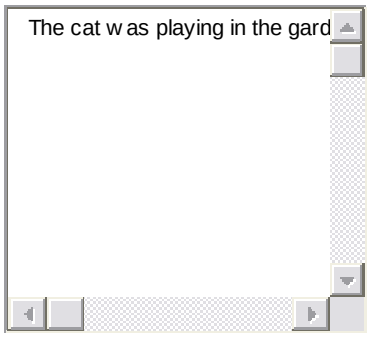
<html><body><form action=""><select name="cars"><option value="volvo">Volvo</option>

<option value="saab">Saab</option><option value="fiat" selected="selected">Fiat</option>

<option value="audi">Audi</option></select></form></body></html>

Textarea
This example demonstrates how to create a text-area (a multi-line text input control). A user can write text in the text-area. In a text-area you can write an unlimited number of characters.

This example cannot be edited because our editor uses a textarea for input, and your browser does not allow a textarea inside a textarea.



Create a button
This example demonstrates how to create a button. On the button you can define your own text.

<html><body><form action=""><input type="button" value="Hello world!"></form></body>

</html>

Fieldset around data
This example demonstrates how to draw a border with a caption around your data.

<html><body><fieldset><legend>Health information:</legend><form action="">

Height <input type="text" size="3">Weight <input type="text" size="3"></form></fieldset>

If there is no border around the input form, your browser is too old. </body></html>

Form Examples

Form with input fields and a submit button
This example demonstrates how to add a form to a page. The form contains two input fields and a submit button.

<html><body><form name="input" action="html_form_action.asp" method="get">

Type your first name: <input type="text" name="FirstName" value="Mickey" size="20">

Type your last name: <input type="text" name="LastName" value="Mouse" size="20">

<input type="submit" value="Submit"></form>

If you click the "Submit" button, you will send your input to a new page called html_form_action.asp. </body></html>

Form with checkboxes
This form contains three checkboxes, and a submit button.

<html><body><form name="input" action="html_form_action.asp" method="get">

I have a bike:<input type="checkbox" name="vehicle" value="Bike" checked="checked" />

I have a car: <input type="checkbox" name="vehicle" value="Car" />

I have an airplane: <input type="checkbox" name="vehicle" value="Airplane" />

<input type="submit" value="Submit" /></form>

If you click the "Submit" button, you send your input to a new page called html_form_action.asp. </body></html>

Form with radio buttons
This form contains two radio buttons, and a submit button.

<html><body><form name="input" action="html_form_action.asp" method="get">

Male: <input type="radio" name="Sex" value="Male" checked="checked">

Female: <input type="radio" name="Sex" value="Female">

<input type ="submit" value ="Submit"></form>

If you click the "Submit" button, you will send your input to a new page called html_form_action.asp. </body></html>

Send e-mail from a form
This example demonstrates how to send e-mail from a form.

<html><body><form action="MAILTO:someone@w3schools.com" method="post" enctype="text/plain"><h3>This form sends an e-mail to W3Schools.</h3>

Name:
<input type="text" name="name"value="yourname" size="20">

Mail:
<input type="text" name="mail"value="yourmail" size="20">

Comment:
<input type="text" name="comment"value="yourcomment" size="40">

<input type="submit" value="Send"><input type="reset" value="Reset"></form></body></html>

Form Tags

Tag	Description
<form>	Defines a form for user input
<input>	Defines an input field
<textarea>	Defines a text-area (a multi-line text input control)
<label>	Defines a label to a control
<fieldset>	Defines a fieldset
<legend>	Defines a caption for a fieldset
<select>	Defines a selectable list (a drop-down box)
<optgroup>	Defines an option group
<option>	Defines an option in the drop-down box
<button>	Defines a push button
<isindex>	Deprecated. Use <input> instead

HTML Images

With HTML you can display images in a document.

Examples

Insert images
This example demonstrates how to display images in your Web page.

<html><body> An image:<img src="constr4.gif"width="144" height="50">

A moving image:<img src="hackanm.gif"width="48" height="48">

Note that the syntax of inserting a moving image is no different from that of a non-moving image.

</body></html>

Insert images from different locations
This example demonstrates how to display images from another folder or another server in your Web page.

<html><body> An image from another folder:<img src="/images/netscape.gif"

width="33" height="32">

An image from W3Schools:<img src="http://www.w3schools.com/images/ie.gif">

</body></html>

The Image Tag and the Src Attribute

In HTML, images are defined with the <img> tag.

The <img> tag is empty, which means that it contains attributes only and it has no closing tag.

To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display on your page.

The syntax of defining an image:

```
<img src="url">
```

The URL points to the location where the image is stored. An image named "boat.gif" located in the directory "images" on "www.w3schools.com" has the URL: http://www.w3schools.com/images/boat.gif.

The browser puts the image where the image tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

The Alt Attribute

The alt attribute is used to define an "alternate text" for an image. The value of the alt attribute is an author-defined text:

```
<img src="boat.gif" alt="Big Boat">
```

The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image. It is a good practice to include the "alt" attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers.

Basic Notes - Useful Tips

If an HTML file contains ten images - eleven files are required to display the page right. Loading images take time, so my best advice is: Use images carefully.

More Examples

Background image
This example demonstrates how to add a background image to an HTML page.

```
<html><body background="background.jpg"><h3>Look: A background image!</h3>

<p>Both gif and jpg files can be used as HTML backgrounds.</p>

<p>If the image is smaller than the page, the image will repeat itself.</p></body></html>
```

Aligning images
This example demonstrates how to align an image within the text.

```
<html><body><p>An image <img src="hackanm.gif"align="bottom" width="48" height="48">

in the text</p>

<p>An image <img src ="hackanm.gif"align="middle" width="48" height="48"> in the text</p>

<p>An image <img src ="hackanm.gif"align="top" width="48" height="48"> in the text</p>

<p>Note that bottom alignment is the default alignment</p>

<p>An image <img src ="hackanm.gif"width="48" height="48"> in the text</p>

<p><img src ="hackanm.gif"width="48" height="48"> An image before the text</p>

<p>An image after the text<img src ="hackanm.gif"width="48" height="48"> </p></body></html>
```

Let the image float
This example demonstrates how to let an image float to the left or right of a paragraph.

```
<html><body><p><img src ="hackanm.gif"align ="left" width="48" height="48">

A paragraph with an image. The align attribute of the image is set to "left". The image will float to the left of this text.</p>

<p><img src ="hackanm.gif"align ="right" width="48" height="48">

A paragraph with an image. The align attribute of the image is set to "right". The image will float to the right of this text.</p></body></html>
```

Adjust images to different sizes
This example demonstrates how to adjust images to different sizes.

```
<html><body><p><img src="hackanm.gif"width="20" height="20"></p>

<p><img src="hackanm.gif"width="45" height="45"></p>

<p><img src="hackanm.gif"width="70" height="70"></p>

<p>You can make a picture larger or smaller changing the values in the "height" and "width"
attributes of the img tag.</p></body></html>
```

Display an alternate text for an image

This example demonstrates how to display an alternate text for an image. The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. It is a good practice to include the "alt" attribute for each image on a page.

```
<html><body><img src="goleft.gif" alt="Go Left" width="32" height="32"><p>

Text-only browsers cannot display images and will only display the text that is specified in the "alt"
attribute for the image. Here, the "alt"-text is "Go Left".</p>

<p>Note that if you hold the mouse pointer over the image, most browsers will display the "alt"-text.

</p></body></html>
```

Make a hyperlink of an image

This example demonstrates how to use an image as a link.

```
<html><body><p>You can also use an image as a link:<a href="lastpage.htm">

<img border="0" src="buttonnext.gif" width="65" height="38"></a></p></body></html>
```

Create an image map

This example demonstrates how to create an image map, with clickable regions. Each of the regions is a hyperlink.

```
<html><body><p>Click on one of the planets to watch it closer:</p>

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

<map id="planetmap" name="planetmap"><area shape="rect" coords="0,0,82,126" alt="Sun"

href="sun.htm">

<area shape="circle" coords="90,58,3" alt="Mercury"href="mercur.htm">

<area shape="circle" coords="124,58,8" alt="Venus"href="venus.htm"></map>

<p><b>Note:</b> The "usemap" attribute in the img element refers to the "id" or "name" (browser
dependant) attribute in the map element, therefore we have added both the "id" and "name" attributes
to the map element.</p></body></html>
```

Turn an image into an image map

This example demonstrates how to turn an image into an image map. You will see that if you move the mouse over the image, the coordinates will be displayed on the status bar.

<html><body> Move the mouse over the image, and look at the status bar to see how the

coordinates change.

<img src="planets.gif"ismap width="146" height="126">

</body></html>

Image Tags

Tag	Description
<img>	Defines an image
<map>	Defines an image map
<area>	Defines a clickable area inside an image map

HTML Backgrounds

A good background can make a Web site look really great.

Examples

Good background and text color
An example of a background color and a text color that makes the text on the page easy to read.

<html><body bgcolor="#d0d0d0">

This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.

This is another paragraph. This is another paragraph. This is another paragraph. This is another paragraph. </body></html>

Bad background and text color
An example of a background color and a text color that makes the text on the page difficult to read.

<html><body bgcolor="#ffffff" text="yellow">

This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.

This is another paragraph. This is another paragraph. This is another paragraph. This is another paragraph. </body></html>

Backgrounds

The <body> tag has two attributes where you can specify backgrounds. The background can be a color or an image.

Bgcolor

The bgcolor attribute specifies a background-color for an HTML page. The value of this attribute can be a hexadecimal number, an RGB value, or a color name:

```
<body bgcolor="#000000">
<body bgcolor="rgb(0,0,0)">
<body bgcolor="black">
```

The lines above all set the background-color to black.

Background

The background attribute specifies a background-image for an HTML page. The value of this attribute is the URL of the image you want to use. If the image is smaller than the browser window, the image will repeat itself until it fills the entire browser window.

```
<body background="clouds.gif">
<body background="http://www.w3schools.com/clouds.gif">
```

The URL can be relative (as in the first line above) or absolute (as in the second line above).

Note: If you want to use a background image, you should keep in mind:

- Will the background image increase the loading time too much?
- Will the background image look good with other images on the page?
- Will the background image look good with the text colors on the page?
- Will the background image look good when it is repeated on the page?
- Will the background image take away the focus from the text?

Basic Notes - Useful Tips

The bgcolor, background, and the text attributes in the <body> tag are deprecated in the latest versions of HTML (HTML 4 and XHTML). The World Wide Web Consortium (W3C) has removed these attributes from its recommendations.

Style sheets (CSS) should be used instead (to define the layout and display properties of HTML elements).

More Examples

Good background image
An example of a background image and a text color that makes the text on the page easy to read.

```
<html><body background="background.jpg"><h3>Image Background</h3>

<p>Both gif and jpg files can be used as HTML backgrounds.</p>

<p>If the image is smaller than the page, the image will repeat itself.</p></body></html>
```

Good background image 2
An example of a background image and a text color that makes the text on the page easy to read.

```
<html><body background="paper.gif"><h3>Image Background</h3>

<p>Both gif and jpg files can be used as HTML backgrounds.</p>

<p>If the image is smaller than the page, the image will repeat itself.</p></body></html>
```

Bad background image
An example of a background image and a text color that makes the text on the page very difficult to read.

```
<html><body background="rock.jpg"><h3>Image Background</h3>

<p>Both gif and jpg files can be used as HTML backgrounds.</p>

<p>If the image is smaller than the page, the image will repeat itself.</p></body></html>
```

Computer Joke

Support: "Type dir, space, a, colon."

Customer: "With a space after 'space'?"

HTML Colors

Colors are displayed combining RED, GREEN, and BLUE light sources.

Color Values

HTML colors can be defined as a hexadecimal notation for the combination of Red, Green, and Blue color values (RGB).

The lowest value that can be given to one light source is 0 (hex #00) and the highest value is 255 (hex #FF).

The table below shows the result of combining Red, Green, and Blue light sources:.

Color	Color HEX	Color RGB
	#000000	rgb(0,0,0)

	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

W3C Standard Color Names

W3C has listed 16 color names that will validate with an HTML validator.

The color names are: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow.

Cross-browser Color Values

Some years ago, when most computers only supported 256 different colors, a list of 216 Web Safe Colors was suggested as a Web standard. The reason for this was that the Microsoft and Mac operating system used 40 different "reserved" fixed system colors (about 20 each).

We are not sure how important this is now, since most computers today have the ability to display millions of different colors, but the choice is left to you.

The 216 cross-browser color palette was created to ensure that all computers would display the colors correctly when running a 256 color palette:

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF

669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

HTML Color Values

Colors are displayed combining RED, GREEN, and BLUE light sources.

Color Values

HTML colors are defined using a hexadecimal notation for the combination of Red, Green, and Blue color values (RGB). The lowest value that can be given to one of the light sources is 0 (hex #00). The highest value is 255 (hex #FF).

Turn Off the Red

If you turn off the Red light completely, there are 65536 different combination of Green and Blue (256 x 256) to experiment with.

Click here to see some of these combinations of Green and Blue.

Turn On the Red

By setting the Red parameter to its maximum value, there are still 65536 different combination of Green and Blue (256 x 256) to experiment with.

16 Million Different Colors

The combination of Red, Green and Blue values from 0 to 255 gives a total of more than 16 million different colors to play with (256 x 256 x 256).

Most modern monitors are capable of displaying at least 16384 different colors.

If you look at the color table below, you will see the result of varying the red light from 0 to 255, while keeping the green and blue light at zero.

To see a full list of 16384 different colors based on red light varying from 0 to 255, click on one of the hexadecimal or rgb values below.

Red Light	HEX	RGB
	#000000	rgb(0,0,0)
	#080000	rgb(8,0,0)
	#100000	rgb(16,0,0)
	#180000	rgb(24,0,0)
	#200000	rgb(32,0,0)
	#280000	rgb(40,0,0)
	#300000	rgb(48,0,0)
	#380000	rgb(56,0,0)
	#400000	rgb(64,0,0)
	#480000	rgb(72,0,0)
	#500000	rgb(80,0,0)
	#580000	rgb(88,0,0)
	#600000	rgb(96,0,0)
	#680000	rgb(104,0,0)
	#700000	rgb(112,0,0)
	#780000	rgb(120,0,0)
	#800000	rgb(128,0,0)
	#880000	rgb(136,0,0)
	#900000	rgb(144,0,0)
	#980000	rgb(152,0,0)
	#A00000	rgb(160,0,0)
	#A80000	rgb(168,0,0)
	#B00000	rgb(176,0,0)
	#B80000	rgb(184,0,0)
	#C00000	rgb(192,0,0)
	#C80000	rgb(200,0,0)
	#D00000	rgb(208,0,0)
	#D80000	rgb(216,0,0)

	#E00000	rgb(224,0,0)
	#E80000	rgb(232,0,0)
	#F00000	rgb(240,0,0)
	#F80000	rgb(248,0,0)
	#FF0000	rgb(255,0,0)

Shades of Gray

Gray colors are displayed using an equal amount of power to all of the light sources. To make it easier for you to select the right gray color we have compiled a table of gray shades for you:

	RGB(0,0,0)	#000000
	RGB(8,8,8)	#080808
	RGB(16,16,16)	#101010
	RGB(24,24,24)	#181818
	RGB(32,32,32)	#202020
	RGB(40,40,40)	#282828
	RGB(48,48,48)	#303030
	RGB(56,56,56)	#383838
	RGB(64,64,64)	#404040
	RGB(72,72,72)	#484848
	RGB(80,80,80)	#505050
	RGB(88,88,88)	#585858
	RGB(96,96,96)	#606060
	RGB(104,104,104)	#686868
	RGB(112,112,112)	#707070
	RGB(120,120,120)	#787878
	RGB(128,128,128)	#808080
	RGB(136,136,136)	#888888
	RGB(144,144,144)	#909090
	RGB(152,152,152)	#989898
	RGB(160,160,160)	#A0A0A0
	RGB(168,168,168)	#A8A8A8
	RGB(176,176,176)	#B0B0B0
	RGB(184,184,184)	#B8B8B8
	RGB(192,192,192)	#C0C0C0
	RGB(200,200,200)	#C8C8C8
	RGB(208,208,208)	#D0D0D0
	RGB(216,216,216)	#D8D8D8
	RGB(224,224,224)	#E0E0E0
	RGB(232,232,232)	#E8E8E8
	RGB(240,240,240)	#F0F0F0
	RGB(248,248,248)	#F8F8F8
	RGB(255,255,255)	#FFFFFF

HTML Color Names

HTML Color Names

The table below provides a list of the color names that are supported by all major browsers.

Note: If you want your pages to validate with an HTML or a CSS validator, W3C has listed 16 color names that you can use: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. If you want to use other colors, you must specify their RGB or HEX value.

Click on a color name (or a hex value) to view the color as the background-color along with different text colors:

Color Name	Color HEX	Color
AliceBlue	#F0F8FF	
AntiqueWhite	#FAEBD7	
Aqua	#00FFFF	
Aquamarine	#7FFFD4	
Azure	#F0FFFF	
Beige	#F5F5DC	
Bisque	#FFE4C4	
Black	#000000	
BlanchedAlmond	#FFEBCD	
Blue	#0000FF	
BlueViolet	#8A2BE2	
Brown	#A52A2A	
BurlyWood	#DEB887	
CadetBlue	#5F9EA0	
Chartreuse	#7FFF00	
Chocolate	#D2691E	
Coral	#FF7F50	
CornflowerBlue	#6495ED	
Cornsilk	#FFF8DC	
Crimson	#DC143C	
Cyan	#00FFFF	
DarkBlue	#00008B	
DarkCyan	#008B8B	
DarkGoldenRod	#B8860B	
DarkGray	#A9A9A9	
DarkGrey	#A9A9A9	
DarkGreen	#006400	
DarkKhaki	#BDB76B	
DarkMagenta	#8B008B	
DarkOliveGreen	#556B2F	

Darkorange	#FF8C00	
DarkOrchid	#9932CC	
DarkRed	#8B0000	
DarkSalmon	#E9967A	
DarkSeaGreen	#8FBC8F	
DarkSlateBlue	#483D8B	
DarkSlateGray	#2F4F4F	
DarkSlateGrey	#2F4F4F	
DarkTurquoise	#00CED1	
DarkViolet	#9400D3	
DeepPink	#FF1493	
DeepSkyBlue	#00BFFF	
DimGray	#696969	
DimGrey	#696969	
DodgerBlue	#1E90FF	
FireBrick	#B22222	
FloralWhite	#FFFAF0	
ForestGreen	#228B22	
Fuchsia	#FF00FF	
Gainsboro	#DCDCDC	
GhostWhite	#F8F8FF	
Gold	#FFD700	
GoldenRod	#DAA520	
Gray	#808080	
Grey	#808080	
Green	#008000	
GreenYellow	#ADFF2F	
HoneyDew	#F0FFF0	
HotPink	#FF69B4	
IndianRed	#CD5C5C	
Indigo	#4B0082	
Ivory	#FFFFF0	
Khaki	#F0E68C	
Lavender	#E6E6FA	
LavenderBlush	#FFF0F5	
LawnGreen	#7CFC00	
LemonChiffon	#FFFACD	
LightBlue	#ADD8E6	
LightCoral	#F08080	
LightCyan	#E0FFFF	
LightGoldenRodYellow	#FAFAD2	
LightGray	#D3D3D3	
LightGrey	#D3D3D3	
LightGreen	#90EE90	

LightPink	#FFB6C1	
LightSalmon	#FFA07A	
LightSeaGreen	#20B2AA	
LightSkyBlue	#87CEFA	
LightSlateGray	#778899	
LightSlateGrey	#778899	
LightSteelBlue	#B0C4DE	
LightYellow	#FFFFE0	
Lime	#00FF00	
LimeGreen	#32CD32	
Linen	#FAF0E6	
Magenta	#FF00FF	
Maroon	#800000	
MediumAquaMarine	#66CDAA	
MediumBlue	#0000CD	
MediumOrchid	#BA55D3	
MediumPurple	#9370D8	
MediumSeaGreen	#3CB371	
MediumSlateBlue	#7B68EE	
MediumSpringGreen	#00FA9A	
MediumTurquoise	#48D1CC	
MediumVioletRed	#C71585	
MidnightBlue	#191970	
MintCream	#F5FFFA	
MistyRose	#FFE4E1	
Moccasin	#FFE4B5	
NavajoWhite	#FFDEAD	
Navy	#000080	
OldLace	#FDF5E6	
Olive	#808000	
OliveDrab	#6B8E23	
Orange	#FFA500	
OrangeRed	#FF4500	
Orchid	#DA70D6	
PaleGoldenRod	#EEE8AA	
PaleGreen	#98FB98	
PaleTurquoise	#AFEEEE	
PaleVioletRed	#D87093	
PapayaWhip	#FFEFD5	
PeachPuff	#FFDAB9	
Peru	#CD853F	
Pink	#FFC0CB	
Plum	#DDA0DD	
PowderBlue	#B0E0E6	

Purple	#800080	
Red	#FF0000	
RosyBrown	#BC8F8F	
RoyalBlue	#4169E1	
SaddleBrown	#8B4513	
Salmon	#FA8072	
SandyBrown	#F4A460	
SeaGreen	#2E8B57	
SeaShell	#FFF5EE	
Sienna	#A0522D	
Silver	#C0C0C0	
SkyBlue	#87CEEB	
SlateBlue	#6A5ACD	
SlateGray	#708090	
SlateGrey	#708090	
Snow	#FFFAFA	
SpringGreen	#00FF7F	
SteelBlue	#4682B4	
Tan	#D2B48C	
Teal	#008080	
Thistle	#D8BFD8	
Tomato	#FF6347	
Turquoise	#40E0D0	
Violet	#EE82EE	
Wheat	#F5DEB3	
White	#FFFFFF	
WhiteSmoke	#F5F5F5	
Yellow	#FFFF00	
YellowGreen	#9ACD32	

HTML 4.01 Quick List

HTML Quick List from W3Schools. Print it, fold it, and put it in your pocket.

HTML Basic Document

```
<html><head>
<title>Document name goes here</title></head>

<body>Visible text goes here</body></html>
```

Heading Elements

```
<h1>Largest Heading</h1>

<h2> . . . </h2>
<h3> . . . </h3>
<h4> . . . </h4>
<h5> . . . </h5>

<h6>Smallest Heading</h6>
```

Text Elements

```
<p>This is a paragraph</p>
<br> (line break)
<hr> (horizontal rule)
<pre>This text is preformatted</pre>
```

Logical Styles

```
<em>This text is emphasized</em>
<strong>This text is strong</strong>
<code>This is some computer code</code>
```

Physical Styles

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

Links, Anchors, and Image Elements

```
<a href="http://www.example.com/">This is a Link</a>
<a href="http://www.example.com/"><img src="URL" alt="Alternate Text"></a>
<a href="mailto:webmaster@example.com">Send e-mail</a>

A named anchor:
<a name="tips">Useful Tips Section</a>
<a href="#tips">Jump to the Useful Tips Section</a>
```

Unordered list

```
<ul>
<li>First item</li>
<li>Next item</li>
</ul>
```

Ordered list

```
<ol>
<li>First item</li>
<li>Next item</li>
</ol>
```

Definition list

```
<dl>
<dt>First term</dt>
<dd>Definition</dd>
<dt>Next term</dt>
```

```
<dd>Definition</dd>
</dl>
```

Tables

```
<table border="1">
<tr>
<th>someheader</th>
<th>someheader</th>
</tr>
<tr>
<td>sometext</td>
<td>sometext</td>
</tr>
</table>
```

Frames

```
<frameset cols="25%,75%">
  <frame src="page1.htm">
  <frame src="page2.htm">
</frameset>
```

Forms

```
<form action="http://www.example.com/test.asp" method="post/get">

<input type="text" name="lastname" value="Nixon" size="30" maxlength="50">
<input type="password">
<input type="checkbox" checked="checked">
<input type="radio" checked="checked">
<input type="submit">
<input type="reset">
<input type="hidden">

<select>
<option>Apples
<option selected>Bananas
<option>Cherries
</select>

<textarea name="Comment" rows="60" cols="20"></textarea>

</form>
```

Entities

```
&lt; is the same as <
&gt; is the same as >
&#169; is the same as ©
```

Other Elements

```
<!-- This is a comment -->
```

```
<blockquote>
Text quoted from some source.
</blockquote>

<address>
Address 1<br>
Address 2<br>
City<br>
</address>
```

Source : http://www.w3schools.com/html/html_quick.asp

HTML Layout

Everywhere on the Web you will find pages that are formatted like newspaper pages using HTML columns.

HTML Layout - Using Tables

One very common practice with HTML, is to use HTML tables to format the layout of an HTML page.

A part of this page is formatted with two columns, like a newspaper page.

As you can see on this page, there is a left column and a right column.

This text is displayed in the left column.

An HTML <table> is used to divide a part of this Web page into two columns.

The trick is to use a table without borders, and maybe a little extra cell-padding.

No matter how much text you add to this page, it will stay inside its column borders.

Same Layout - Color Added

One very common practice with HTML, is to use HTML tables to format the layout of an HTML page.

A part of this page is formatted with two columns, like a newspaper page.

As you can see at this page, there is a left column and a right column.

An HTML <table> is used to divide a part of this Web page into two columns.

This text is displayed in the right column.

The trick is to use a table without borders, and maybe a little extra cell-padding.

No matter how much text you add to this page, it will stay inside its column borders.

Examples

Dividing a part of an HTML page into table columns is very easy to do. To let you experiment with it, we have put together this simple example.

<html><body><table border="0" width="100%" cellpadding="10"><tr>

<td width="50%" valign="top">This is some text. This is some text. This is some text. This is some text. This is some text.</td><td width="50%" valign="top">

Another text. Another text. Another text. Another text. Another text. Another text. Another text.

</td></tr></table></body></html>

HTML Joke

Student: "How do you spell HTML?"

HTML Fonts

The <font> tag in HTML is deprecated. It is supposed to be removed in a future version of HTML.

Even if a lot of people are using it, you should try to avoid it, and use styles instead.

The HTML <font> Tag

With HTML code like this, you can specify both the size and the type of the browser output :

```
<p>
<font size="2" face="Verdana">
This is a paragraph.
</font>
</p>
<p>
<font size="3" face="Times">
This is another paragraph.
</font>
</p>
```

Font Attributes

Attribute	Example	Purpose
size="number"	size="2"	Defines the font size
size="+number"	size="+1"	Increases the font size
size="-number"	size="-1"	Decreases the font size
face="face-name"	face="Times"	Defines the font-name

color="color-value"	color="#eeff00"	Defines the font color
color="color-name"	color="red"	Defines the font color

The <font> Tag Should NOT be Used

The <font> tag is deprecated in the latest versions of HTML (HTML 4 and XHTML).

The World Wide Web Consortium (W3C) has removed the <font> tag from its recommendations. In future versions of HTML, style sheets (CSS) will be used to define the layout and display properties of HTML elements.

The Right Way to Do It - With Styles

Set the font of text
This example demonstrates how to set the font of a text.

```
<html><body><h1 style="font-family:verdana">A heading</h1>

<p style="font-family:courier">A paragraph</p></body></html>
```

Set the font size of text
This example demonstrates how to set the font size of a text.

```
<html><body><h1 style="font-size:150%">A heading</h1>

<p style="font-size:80%">A paragraph</p></body></html>
```

Set the font color of text
This example demonstrates how to set the color of a text.

```
<html><body><h1 style="color:blue">A heading</h1><p style="color:red">A paragraph</p>

</body></html>
```

Set the font, font size, and font color of text
This example demonstrates how to set the font, font size, and font color of a text.

```
<html><body><p style="font-family:verdana;font-size:80%;color:green">

This is a paragraph with some text in it. This is a paragraph with some text in it. This is a paragraph with some text in it. This is a paragraph with some text in it.</p></body></html>
```

Where to Learn More About Style Sheets?

First off: Finish the last chapters in our HTML tutorial !!! In the following chapters we will explain why some tags, like <font>, are to be removed from the HTML recommendations, and how to insert a style sheet in an HTML document.

To learn more about style sheets: Study our CSS Tutorial.

Why use HTML 4.0?

HTML 3.2 Was Very Wrong !

The original HTML was **never intended** to contain tags for **formatting** a document. HTML tags were intended to define the **content** of the document like:

This is a paragraph

<h1>This is a heading</h1>

When tags like <font> and color attributes were added to the HTML 3.2 specification, it started a **nightmare** for web developers. Development of large web sites where fonts and color information had to be added to every single Web page, became a long, expensive and unduly painful process.

What is so Great About HTML 4.0 ?

In HTML 4.0 **all formatting can be removed** from the HTML document and stored in a separate style sheet.

Because HTML 4.0 separates the presentation from the document structure, we have what we always needed: Total control of presentation layout without messing up the document content.

What Should You do About it ?

Do not use presentation attributes inside your HTML tags if you can avoid it. Start using styles! Please read our CSS tutorial to learn about style sheets.

Do not use deprecated tags. Visit our complete HTML 4.01 Reference to see which tags and attributes that are deprecated.

Prepare Yourself for XHTML

XHTML is the "new" HTML. The most important thing you can do is to start writing valid HTML 4.01. Also start writing your tags in lower case. Always close your tag elements. Never end a paragraph without .

NOTE: The official HTML 4.01 recommends the use of lower case tags.

If you want to read about how this web site was converted to XHTML, please visit our XHTML tutorial.

Validate Your HTML Files as HTML 4.01

An HTML document is validated against a Document Type Definition (DTD). Before an HTML file can be properly validated, a correct DTD must be added as the first line of the file.

The HTML 4.01 Strict DTD includes elements and attributes that have not been deprecated or do not appear in framesets:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

The HTML 4.01 Transitional DTD includes everything in the strict DTD plus deprecated elements and attributes:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

The HTML 4.01 Frameset DTD includes everything in the transitional DTD plus frames as well:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

HTML Styles

With HTML 4.0 all formatting can be moved out of the HTML document and into a separate style sheet.

Examples

Styles in HTML
This example demonstrates how to format an HTML document with style information added to the <head> section.

```
<html><head><style type="text/css">h1 {color: red}h3 {color: blue}</style></head>

<body><h1>This is header 1</h1><h3>This is header 3</h3></body></html>
```

Link that is not underlined
This example demonstrates how to make a link that is not underlined, using a style attribute.

```
<html><body><a href="lastpage.htm" style="text-decoration:none">THIS IS A LINK!</a>

</body></html>
```

Link to an external style sheet
This example demonstrates how to use the <link> tag to link to an external style sheet.

<html><head><link rel="stylesheet" type="text/css" href="styles.css" ></head><body>

<h1>I am formatted with a linked style sheet</h1> Me too! </body></html>

How to Use Styles

When a browser reads a style sheet, it will format the document according to it. There are three ways of inserting a style sheet:

External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section.

```
<head>
<link rel="stylesheet" type="text/css"
href="mystyle.css">
</head>
```

Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section with the <style> tag.

```
<head>
<style type="text/css">
body {background-color: red}
p {margin-left: 20px}
</style>
</head>
```

Inline Styles

An inline style should be used when a unique style is to be applied to a single occurrence of an element.

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color: red; margin-left: 20px">
This is a paragraph
</p>
```

To learn more about styles, visit our CSS tutorial.

Style Tags

Tag	Description
<style>	Defines a style definition
<link>	Defines a resource reference
	Defines a section in a document
	Defines a section in a document
<font>	Deprecated. Use styles instead
<basefont>	Deprecated. Use styles instead
	Deprecated. Use styles instead

Joke

Customer: Hello, it's me!
Support: It's me too!
Customer: No, Esmie. E, s, m, i, e!
Support: Sorry!

HTML Head

Examples

The title of a document
The title information inside a head element is not displayed in the browser window.

<html><head><title>The title is not displayed</title></head><body> This text is displayed

</body></html>

One target for all links
This example demonstrates how to use the base tag to let all the links on a page open in a new window.

<html><head><base target="_blank"></head><body> <a href="http://www.w3schools.com"

target="_blank">This link

will load in a new window because the target attribute is set to "_blank".

This link

will also load in a new window even without a target attribute. </body></html>

The Head Element

The head element contains general information, also called meta-information, about a document. Meta means "information about".

You can say that meta-data means information about data, or meta-information means information about information.

Information Inside the Head Element

The elements inside the head element should not be displayed by a browser.

According to the HTML standard, only a few tags are legal inside the head section. These are: <base>, <link>, <meta>, <title>, <style>, and <script>.

Look at the following illegal construct:

```
<head>
  <p>This is some text</p>
</head>
```

In this case the browser has two options:

- Display the text because it is inside a paragraph element
- Hide the text because it is inside a head element

If you put an HTML element like <h1> or inside a head element like this, most browsers will display it, even if it is illegal.

Should browsers forgive you for errors like this? We don't think so. Others do.

Head Tags

Tag	Description
<head>	Defines information about the document
<title>	Defines the document title
<base>	Defines a base URL for all the links on a page
<link>	Defines a resource reference
<meta>	Defines meta information

Tag	Description
<!DOCTYPE>	Defines the document type. This tag goes before the <html> start tag.

HTML Meta

Examples

Document description
Information inside a meta element describes the document.

```
<html><head><meta name="author"content="Jan Egil Refsnes"><meta name="revised"

content="Jan Egil Refsnes,6/10/99"><meta name="generator"content="Microsoft FrontPage 4.0">

</head><body><p>The meta attributes of this document identify the author and the editor software.

</p></body></html>
```

Document keywords
Information inside a meta element describes the document's keywords.

```
<html><head><meta name="description"content="HTML examples"><meta name="keywords"

content="HTML, DHTML, CSS, XML, XHTML, JavaScript, VBScript"></head><body><p>

The meta attributes of this document describe the document and its keywords.</p></body></html>
```

Redirect a user
This example demonstrates how to redirect a user if your site address has changed.

```
<html><head><meta http-equiv="Refresh"content="5;url=http://www.w3schools.com"></head>

<body><p>

Sorry! We have moved! The new URL is: <a
href="http://www.w3schools.com">http://www.w3schools.com</a></p>

<p>You will be redirected to the new address in five seconds.</p>

<p>If you see this message for more than 5 seconds, please click on the link above!</p>

</body></html>
```

The Meta Element

As we explained in the previous chapter, the head element contains general information (meta-information) about a document.

HTML also includes a meta element that goes inside the head element. The purpose of the meta element is to provide meta-information about the document.

Most often the meta element is used to provide information that is relevant to browsers or search engines like describing the content of your document.

Note: W3C states that "*Some user agents support the use of META to refresh the current page after a specified number of seconds, with the option of replacing it by a different URI. Authors should not use this technique to forward users to different pages, as this makes the page inaccessible to some users. Instead, automatic page forwarding should be done using server-side redirects*" at http://www.w3.org/TR/html4/struct/global.html#adef-http-equiv.

Keywords for Search Engines

Some search engines on the WWW will use the name and content attributes of the meta tag to index your pages.

This meta element defines a description of your page:

```
<meta name="description" content="Free Web tutorials on HTML, CSS, XML, and XHTML">
```

This meta element defines keywords for your page:

```
<meta name="keywords" content="HTML, DHTML, CSS, XML, XHTML, JavaScript, VBScript">
```

The intention of the name and content attributes is to describe the content of a page.

However, since too many webmasters have used meta tags for spamming, like repeating keywords to give pages a higher ranking, some search engines have stopped using them entirely.

You can read more about search engines in our Web Building Tutorial.

Unknown Meta Attributes

Sometimes you will see meta attributes that are unknown to you like this:

```
<meta name="security" content="low">
```

Then you just have to accept that this is something unique to the site or to the author of the site, and that it has probably no relevance to you.

HTML Uniform Resource Locators

HTML Links

When you click on a link in an HTML document like this: Last Page, an underlying <a> tag points to a place (an address) on the Web with an href attribute value like this: Last Page.

The Last Page link in the example is a link that is relative to the Web site that you are browsing, and your browser will construct a full Web address like http://www.w3schools.com/html/lastpage.htm to access the page.

Uniform Resource Locators

Something called a Uniform Resource Locator (URL) is used to address a document (or other data) on the World Wide Web. A full Web address like this: http://www.w3schools.com/html/lastpage.htm follows these syntax rules:

scheme://host.domain:port/path/filename

The **scheme** is defining the **type** of Internet service. The most common type is **http**.

The **domain** is defining the Internet **domain name** like w3schools.com.

The **host** is defining the domain host. If omitted, the default host for http is **www**.

The **:port** is defining the **port number** at the host. The port number is normally omitted. The default port number for http is **80**.

The **path** is defining a **path** (a sub directory) at the server. If the path is omitted, the resource (the document) must be located at the root directory of the Web site.

The **filename** is defining the name of a document. The default filename might be default.asp, or index.html or something else depending on the settings of the Web server.

URL Schemes

Some examples of the most common schemes can be found below:

Schemes	Access
file	a file on your local PC
ftp	a file on an FTP server
http	a file on a World Wide Web Server
gopher	a file on a Gopher server
news	a Usenet newsgroup
telnet	a Telnet connection
WAIS	a file on a WAIS server

Accessing a Newsgroup

The following HTML code:

HTML Newsgroup

creates a link to a newsgroup like this HTML Newsgroup.

Downloading with FTP

The following HTML code:

Download WinZip

creates a link to download a file like this: Download WinZip.

(The link doesn't work. Don't try it. It is just an example. W3Schools doesn't really have an ftp directory.)

Link to your Mail system

The following HTML code:

someone@w3schools.com

creates a link to your own mail system like this:

someone@w3schools.com

HTML Scripts

Add scripts to HTML pages to make them more dynamic and interactive.

Examples

Insert a script
This example demonstrates how to insert a script into your HTML document.

Work with browsers that do not support scripts
This example demonstrates how to handle browsers that do not support scripting.

<html><body><script type="text/javascript"><!--

document.write("If this is displayed, your browser supports scripting!")

//-->

</script><noscript>No JavaScript support!</noscript>

A browser that does not support JavaScript will show the text in the noscript element.

</body></html>

Insert a Script into HTML Page

A script in HTML is defined with the <script> tag. Note that you will have to use the type attribute to specify the scripting language.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

The script above will produce this output:

Hello World!

Note: To learn more about scripting in HTML, visit our JavaScript School.

How to Handle Older Browsers

A browser that does not recognize the <script> tag at all, will display the <script> tag's content as text on the page. To prevent the browser from doing this, you should hide the script in comment tags. An old browser (that does not recognize the <script> tag) will ignore the comment and it will not write the tag's content on the page, while a new browser will understand that the script should be executed, even if it is surrounded by comment tags.

Example

```
JavaScript:
<script type="text/javascript">
<!--
document.write("Hello World!")
//-->
</script>

VBScript:
<script type="text/vbscript">
<!--
document.write("Hello World!")
'-->
</script>
```

The <noscript> Tag

In addition to hiding the script inside a comment, you can also add a <noscript> tag.

The <noscript> tag is used to define an alternate text if a script is NOT executed. This tag is used for browsers that recognize the <script> tag, but do not support the script inside, so these browsers will display the text inside the <noscript> tag instead. However, if a browser supports the script inside the <script> tag it will ignore the <noscript> tag.

Example

```
JavaScript:
<script type="text/javascript">
<!--
document.write("Hello World!")
//-->
</script>
<noscript>Your browser does not support JavaScript!</noscript>

VBScript:
<script type="text/vbscript">
<!--
document.write("Hello World!")
'-->
</script>
<noscript>Your browser does not support VBScript!</noscript>
```

Script Tags

Tag	Description
<script>	Defines a script
<noscript>	Defines an alternate text if the script is not executed
<object>	Defines an embedded object
<param>	Defines run-time settings (parameters) for an object
<applet>	Deprecated. Use <object> instead

HTML 4.0 Standard Attributes

HTML tags can have attributes. The special attributes for each tag are listed under each tag description. The attributes listed here are the core and language attributes that are standard for all tags (with a few exceptions):

Core Attributes

Not valid in base, head, html, meta, param, script, style, and title elements.

Attribute	Value	Description
class	*class_rule* or *style_rule*	The class of the element
id	*id_name*	A unique id for the element
style	*style_definition*	An inline style definition
title	*tooltip_text*	A text to display in a tool tip

Language Attributes

Not valid in base, br, frame, frameset, hr, iframe, param, and script elements.

Attribute	Value	Description
dir	ltr | rtl	Sets the text direction
lang	*language_code*	Sets the language code

Keyboard Attributes

Attribute	Value	Description
accesskey	*character*	Sets a keyboard shortcut to access an element
tabindex	*number*	Sets the tab order of an element

HTML 4.0 Event Attributes

New to HTML 4.0 is the ability to let HTML events trigger actions in the browser, like starting a JavaScript when a user clicks on an HTML element. Below is a list of attributes that can be inserted into HTML tags to define event actions.

If you want to learn more about programming with these events, you should study our JavaScript tutorial and our DHTML tutorial.

Window Events

Only valid in body and frameset elements.

Attribute	Value	Description
onload	*script*	Script to be run when a document loads
onunload	*script*	Script to be run when a document unloads

Form Element Events

Only valid in form elements.

Attribute	Value	Description
onchange	*script*	Script to be run when the element changes
onsubmit	*script*	Script to be run when the form is submitted
onreset	*script*	Script to be run when the form is reset
onselect	*script*	Script to be run when the element is selected
onblur	*script*	Script to be run when the element loses focus
onfocus	*script*	Script to be run when the element gets focus

---

Keyboard Events

Not valid in base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, and title elements.

Attribute	Value	Description
onkeydown	*script*	What to do when key is pressed
onkeypress	*script*	What to do when key is pressed and released
onkeyup	*script*	What to do when key is released

Mouse Events

Not valid in base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, title elements.

Attribute	Value	Description
onclick	*script*	What to do on a mouse click
ondblclick	*script*	What to do on a mouse double-click
onmousedown	*script*	What to do when mouse button is pressed
onmousemove	*script*	What to do when mouse pointer moves
onmouseout	*script*	What to do when mouse pointer moves out of an element
onmouseover	*script*	What to do when mouse pointer moves over an element
onmouseup	*script*	What to do when mouse button is released

HTML URL-encoding Reference

Below is a reference of ASCII characters in URL-encoding form (hexadecimal format).

Hexadecimal values can be used to display non-standard letters and characters in browsers and plug-ins.

Try It

Type some text or an ASCII value in the input field below, and click on the "URL Encode" button to see the URL-encoding.

URL Encode

URL-encoding from %00 to %8f

ASCII Value	URL-encode	ASCII Value	URL-encode	ASCII Value	URL-encode
æ	%00	0	%30	`	%60
	%01	1	%31	a	%61
	%02	2	%32	b	%62
	%03	3	%33	c	%63
	%04	4	%34	d	%64
	%05	5	%35	e	%65
	%06	6	%36	f	%66
	%07	7	%37	g	%67
backspace	%08	8	%38	h	%68
tab	%09	9	%39	i	%69
linefeed	%0a	:	%3a	j	%6a
	%0b	;	%3b	k	%6b
	%0c	<	%3c	l	%6c
c return	%0d	=	%3d	m	%6d
	%0e	>	%3e	n	%6e
	%0f	?	%3f	o	%6f
	%10	@	%40	p	%70
	%11	A	%41	q	%71
	%12	B	%42	r	%72
	%13	C	%43	s	%73
	%14	D	%44	t	%74
	%15	E	%45	u	%75
	%16	F	%46	v	%76
	%17	G	%47	w	%77
	%18	H	%48	x	%78
	%19	I	%49	y	%79
	%1a	J	%4a	z	%7a

	%1b	K	%4b	{	%7b
	%1c	L	%4c	|	%7c
	%1d	M	%4d	}	%7d
	%1e	N	%4e	~	%7e
	%1f	O	%4f		%7f
space	%20	P	%50	€	%80
!	%21	Q	%51		%81
"	%22	R	%52	‚	%82
#	%23	S	%53	ƒ	%83
$	%24	T	%54	„	%84
%	%25	U	%55	…	%85
&	%26	V	%56	†	%86
'	%27	W	%57	‡	%87
(	%28	X	%58	ˆ	%88
)	%29	Y	%59	‰	%89
*	%2a	Z	%5a	Š	%8a
+	%2b	[	%5b	‹	%8b
,	%2c	\	%5c	Œ	%8c
-	%2d	]	%5d		%8d
.	%2e	^	%5e	Ž	%8e
/	%2f	_	%5f		%8f

URL-encoding from %90 to %ff

ASCII Value	URL-encode	ASCII Value	URL-encode	ASCII Value	URL-encode
	%90	À	%c0	ð	%f0
‘	%91	Á	%c1	ñ	%f1
’	%92	Â	%c2	ò	%f2
“	%93	Ã	%c3	ó	%f3
”	%94	Ä	%c4	ô	%f4
•	%95	Å	%c5	õ	%f5
–	%96	Æ	%c6	ö	%f6
—	%97	Ç	%c7	÷	%f7
˜	%98	È	%c8	ø	%f8
™	%99	É	%c9	ù	%f9
š	%9a	Ê	%ca	ú	%fa
›	%9b	Ë	%cb	û	%fb
œ	%9c	Ì	%cc	ü	%fc
	%9d	Í	%cd	ý	%fd
ž	%9e	Î	%ce	þ	%fe
Ÿ	%9f	Ï	%cf	ÿ	%ff
	%a0	Đ	%d0		
¡	%a1	Ñ	%d1		
¢	%a2	Ò	%d2		
£	%a3	Ó	%d3		

	%a4	Ô	%d4		
¥	%a5	Õ	%d5		
|	%a6	Ö	%d6		
§	%a7		%d7		
¨	%a8	Ø	%d8		
©	%a9	Ù	%d9		
a	%aa	Ú	%da		
«	%ab	Û	%db		
¬	%ac	Ü	%dc		
¯	%ad	Ý	%dd		
®	%ae	Þ	%de		
¯	%af	ß	%df		
°	%b0	à	%e0		
±	%b1	á	%e1		
²	%b2	â	%e2		
³	%b3	ã	%e3		
´	%b4	ä	%e4		
µ	%b5	å	%e5		
¶	%b6	æ	%e6		
·	%b7	ç	%e7		
¸	%b8	è	%e8		
¹	%b9	é	%e9		
º	%ba	ê	%ea		
»	%bb	ë	%eb		
¼	%bc	ì	%ec		
½	%bd	í	%ed		
¾	%be	î	%ee		
¿	%bf	ï	%ef		

Ready to Publish Your Work?

Your First Step: A Personal Web Server

- If you want other people to view your pages, you must publish them.
- To publish your work, you have to copy your files to a web server.
- Your own PC can act as a web server if it is connected to a network.
- If you are running Windows 98, you can use the PWS (Personal Web Server).
- PWS is hiding in the PWS folder in your Windows CD.

Personal Web Server (PWS)

PWS turns any Windows computer into a Web server. PWS is easy to install and ideal for developing and testing Web applications. PWS has been optimized for workstation use, but has all the

requirements of a full Web server. It also runs Active Server Pages (ASP) just like its larger brother IIS.

How to Install a Personal Web Server (PWS):

- Browse your Windows installation to see if you have installed PWS.
- If not, install PWS from the PWS directory on your Windows CD.
- Follow the instructions and get your Personal Web Server up and running.

Note: Microsoft Windows XP Home Edition does not come with the option to turn your computer into a PWS!

Internet Information Server (IIS)

Windows 2000's built-in Web server IIS, makes it easy to build large applications for the Web. Both PWS and IIS include ASP, a server-side scripting standard that can be used to create dynamic and interactive Web applications. IIS is also available for Windows NT.

Your Next Step: A Professional Web Server

- If you do not want to use PWS or IIS, you must upload your files to a public server.
- Most Internet Service Providers (ISP's) will offer to host your web pages.
- If your employer has an Internet Server, you can ask him to host your Web site.
- If you are really serious about this, you should install your own Internet Server.

Before you select an ISP, make sure you read W3Schools Web Hosting Tutorial !!

You Have Learned HTML, Now What?

HTML Summary

This tutorial has taught you how to use HTML to create your own web site.

HTML is the universal markup language for the Web. HTML lets you format text, add graphics, create links, input forms, frames and tables, etc., and save it all in a text file that any browser can read and display.

The key to HTML is the tags, which indicates what content is coming up.

Now You Know HTML, What's Next?

The next step is to learn XHTML and CSS.

XHTML

XHTML reformulates HTML 4.01 in XML.

XHTML Tutorial

XHTML Tutorial

XHTML is a stricter and cleaner version of HTML.

In this tutorial you will learn the difference between HTML and XHTML. We will also show you how this Web site was converted to XHTML.

Start learning XHTML now!

Introduction To XHTML

XHTML is a stricter and cleaner version of HTML.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML and the basics of building web pages

What Is XHTML?

- XHTML stands for E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
- XHTML is aimed to **replace** HTML
- XHTML is almost **identical** to HTML 4.01
- XHTML is a **stricter and cleaner** version of HTML
- XHTML is HTML defined as an **XML application**
- XHTML is a W3C Recommendation

XHTML is a W3C Recommendation

XHTML 1.0 became a W3C Recommendation January 26, 2000.

W3C defines XHTML as the latest version of HTML. XHTML will gradually replace HTML.

All New Browsers Support XHTML

XHTML is compatible with HTML 4.01.

All new browsers have support for XHTML.

About This Tutorial

The next chapters of this tutorial will explain:

- Why you should use XHTML
- The syntax of XHTML
- How W3Schools was converted to XHTML
- XHTML validation
- XHTML modularization

XHTML - Why?

- **XHTML is a combination of HTML and XML (EXtensible Markup Language).**
- **XHTML consists of all the elements in HTML 4.01 combined with the syntax of XML.**
- **Why XHTML?**
- We have reached a point where many pages on the WWW contain "bad" HTML.
- The following HTML code will work fine if you view it in a browser, even if it does not follow the HTML rules:

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

- XML is a markup language where everything has to be marked up correctly, which results in "well-formed" documents.
- XML was designed to describe data and HTML was designed to display data.
- Today's market consists of different browser technologies, some browsers run Internet on computers, and some browsers run Internet on mobile phones and hand helds. The last-mentioned do not have the resources or power to interpret a "bad" markup language.
- Therefore - by combining HTML and XML, and their strengths, we got a markup language that is useful now and in the future - XHTML.
- XHTML pages can be read by all XML enabled devices AND while waiting for the rest of the world to upgrade to XML supported browsers, XHTML gives you the opportunity to write "well-formed" documents now, that work in all browsers and that are backward browser compatible !!!

Differences Between XHTML And HTML

You can prepare yourself for XHTML by starting to write strict HTML.

How To Get Ready For XHTML

XHTML is not very different from the HTML 4.01 standard.

So, bringing your code up to the 4.01 standard is a good start. Our complete HTML 4.01 reference can help you with that.

In addition, you should start NOW to write your HTML code in lowercase letters, and NEVER skip ending tags (like </p>).

Happy coding!

The Most Important Differences:

- 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 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>
```

Note: A common mistake with nested lists, is to forget that the inside list must be within <li> and tags.

This is wrong:

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  <li>Milk</li>
</ul>
```

This is correct:

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Notice that we have inserted a tag after the tag in the "correct" code example.

XHTML Elements Must Always Be Closed

Non-empty elements must have an end tag.

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

Empty elements must either have an end tag or the start tag must end with />.

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" />
```

XHTML Elements Must Be In Lower Case

The XHTML specification defines that the tag names and attributes need to be lower case.

This is wrong:

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

This is correct:

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

XHTML Documents Must Have One Root Element

All XHTML elements must be nested within the <html> root element. All other elements can have sub (children) elements. Sub elements must be in pairs and correctly nested within their parent element. The basic document structure is:

```
<html>
<head> ... </head>
<body> ... </body>
</html>
```

XHTML Syntax

Writing XHTML demands a clean HTML syntax.

Some More XHTML Syntax Rules:

- Attribute names must be in **lower case**
- Attribute values must be **quoted**
- Attribute minimization is **forbidden**
- The id attribute **replaces** the name attribute
- The XHTML DTD defines **mandatory** elements

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

This is wrong:

```
<input checked>
<input readonly>
<input disabled>
<option selected>
<frame noresize>
```

This is correct:

```
<input checked="checked" />
<input readonly="readonly" />
<input disabled="disabled" />
<option selected="selected" />
<frame noresize="noresize" />
```

Here is a list of the minimized attributes in HTML and how they should be written in XHTML:

HTML	XHTML
compact	compact="compact"
checked	checked="checked"
declare	declare="declare"
readonly	readonly="readonly"
disabled	disabled="disabled"
selected	selected="selected"
defer	defer="defer"
ismap	ismap="ismap"
nohref	nohref="nohref"
noshade	noshade="noshade"
nowrap	nowrap="nowrap"
multiple	multiple="multiple"
noresize	noresize="noresize"

The id Attribute Replaces The name Attribute

HTML 4.01 defines a name attribute for the elements a, applet, frame, iframe, img, and map. In XHTML the name attribute is deprecated. Use id instead.

This is wrong:

```
<img src="picture.gif" name="picture1" />
```

This is correct:

```
<img src="picture.gif" id="picture1" />
```

Note: To interoperate with older browsers for a while, you should use both name and id, with identical attribute values, like this:

```
<img src="picture.gif" id="picture1" name="picture1" />
```

IMPORTANT Compatibility Note:

To make your XHTML compatible with today's browsers, you should add an extra space before the "/" symbol.

The Lang Attribute

The lang attribute applies to almost every XHTML element. It specifies the language of the content within an element.

If you use the lang attribute in an element, you must add the xml:lang attribute, like this:

```
<div lang="no" xml:lang="no">Heia Norge!</div>
```

Mandatory XHTML Elements

All XHTML documents must have a DOCTYPE declaration. The html, head and body elements must be present, and the title must be present inside the head element.

This is a minimum XHTML document template:

```
<!DOCTYPE Doctype goes here>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Title goes here</title>
</head>
<body>
</body>
</html>
```

Note: The DOCTYPE declaration is not a part of the XHTML document itself. It is not an XHTML element, and it should not have a closing tag.

XHTML DTD

The XHTML standard defines three Document Type Definitions.

The most common is the XHTML Transitional.

<!DOCTYPE> Is Mandatory

An XHTML document consists of three main parts:

- the DOCTYPE
- the Head
- the Body

The basic document structure is:

```
<!DOCTYPE ...>
<html>
<head>
<title>... </title>
</head>
<body> ... </body>
</html>
```

The DOCTYPE declaration should always be the first line in an XHTML document.

An XHTML Example

This is a simple (minimal) XHTML document:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>simple document</title>
</head>
<body>
<p>a simple paragraph</p>
</body>
</html>
```

The DOCTYPE declaration defines the document type:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

The rest of the document looks like HTML:

```
<html>
<head>
<title>simple document</title>
</head>
<body>
<p>a simple paragraph</p>
</body>
</html>
```

The 3 Document Type Definitions

- DTD specifies the syntax of a web page in SGML.
- DTD is used by SGML applications, such as HTML, to specify rules that apply to the markup of documents of a particular type, including a set of element and entity declarations.
- XHTML is specified in an SGML document type definition or 'DTD'.
- An XHTML DTD describes in precise, computer-readable language, the allowed syntax and grammar of XHTML markup.

There are currently 3 XHTML document types:

- STRICT
- TRANSITIONAL
- FRAMESET

XHTML 1.0 specifies three XML document types that correspond to three DTDs: Strict, Transitional, and Frameset.

XHTML 1.0 Strict

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Use this when you want really clean markup, free of presentational clutter. Use this together with Cascading Style Sheets.

XHTML 1.0 Transitional

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Use this when you need to take advantage of HTML's presentational features and when you want to support browsers that don't understand Cascading Style Sheets.

XHTML 1.0 Frameset

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Use this when you want to use HTML Frames to partition the browser window into two or more frames.

XHTML HowTo

How W3Schools Was Converted To XHTML

W3Schools was converted from HTML to XHTML the weekend of 18. and 19. December 1999, by Hege Refsnes and Ståle Refsnes.

To convert a Web site from HTML to XHTML, you should be familiar with the XHTML syntax rules of the previous chapters. The following steps were executed (in the order listed below):

A DOCTYPE Definition Was Added

The following DOCTYPE declaration was added as the first line of every page:

```
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Note that we used the transitional DTD. We could have chosen the strict DTD, but found it a little too "strict", and a little too hard to conform to.

A Note About The DOCTYPE

Your pages must have a DOCTYPE declaration if you want them to validate as correct XHTML.

Be aware however, that newer browsers (like Internet Explorer 6) might treat your document differently depending on the <!DOCTYPE> declaration. If the browser reads a document with a DOCTYPE, it might treat the document as "correct". Malformed XHTML might fall over and display differently than without a DOCTYPE.

Lower Case Tag And Attribute Names

Since XHTML is case sensitive, and since XHTML only accepts lower case HTML tags and attribute names, a general search and replace function was executed to replace all upper case tags with lowercase tags. The same was done for attribute names. We have always tried to use lower case names in our Web, so the replace function did not produce many real substitutions.

All Attributes Were Quoted

Since the W3C XHTML 1.0 Recommendation states that all attribute values must be quoted, every page in the web was checked to see that attributes values were properly quoted. This was a time-consuming job, and we will surely never again forget to put quotes around our attribute values.

**Empty Tags: <hr> ,
 and <img>**

Empty tags are not allowed in XHTML. The <hr> and
 tags should be replaced with <hr /> and
.

This produced a problem with Netscape that misinterpreted the
 tag. We don't know why, but changing it to
 worked fine. After that discovery, a general search and replace function was executed to swap the tags.

A few other tags (like the <img> tag) were suffering from the same problem as above. We decided not to close the <img> tags with </img>, but with /> at the end of the tag. This was done manually.

The Web Site Was Validated

After that, all pages were validated against the official W3C DTD with this link: XHTML Validator. A few more errors were found and edited manually. The most common error was missing tags in lists.

Should we have used a converting tool? Well, we could have used **TIDY**.

Dave Raggett's HTML TIDY is a free utility for cleaning up HTML code. It also works great on the hard-to-read markup generated by specialized HTML editors and conversion tools, and it can help you identify where you need to pay further attention on making your pages more accessible to people with disabilities.

The reason why we didn't use Tidy? We knew about XHTML when we started writing this web site. We knew that we had to use lowercase tag names and that we had to quote our attributes. So when the time came (to do the conversion), we simply had to test our pages against the W3C XHTML validator and correct the few mistakes. AND - we have learned a lot about writing "tidy" HTML code.

XHTML Validation

An XHTML document is validated against a Document Type Definition.

Validate XHTML With A DTD

An XHTML document is validated against a Document Type Definition (DTD). Before an XHTML file can be properly validated, a correct DTD must be added as the first line of the file.

The Strict DTD includes elements and attributes that have not been deprecated or do not appear in framesets:

!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"

The Transitional DTD includes everything in the strict DTD plus deprecated elements and attributes:

!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"

The Frameset DTD includes everything in the transitional DTD plus frames as well:

!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Frameset//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd"

This is a simple XHTML document:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>simple document</title>
</head>
<body>
<p>a simple paragraph</p>
</body>
</html>
```

XHTML Modularization

The XHTML modularization-model defines the modules of XHTML.

Why XHTML Modularization?

XHTML is a simple, but large language. XHTML contains most of the functionality a web developer will need.

For some purposes XHTML is too large and complex, and for other purposes it's too simple.

By splitting XHTML into modules, the W3C (World Wide web Consortium) has created small and well-defined sets of XHTML elements that can be used separately for small devices, or combined with other XML standards into larger and more complex applications.

With modular XHTML, designers can:

- Choose the elements to be supported by a device using standard XHTML building blocks
- Add extensions to XHTML, using XML, without breaking the XHTML standard
- Simplify XHTML for devices like hand held computers, mobile phones, TV, and home appliances
- Extend XHTML for complex applications by adding new XML functionality (like MathML, SVG, Voice and Multimedia)
- Define XHTML profiles like XHTML Basic (a subset of XHTML for mobile devices)

XHTML Modules

W3C has split the definition of XHTML into 28 modules:

Module name	Description
Applet Module	Defines the deprecated* applet element
Base Module	Defines the base element
Basic Forms Module	Defines the basic forms elements
Basic Tables Module	Defines the basic table elements
Bi-directional Text Module	Defines the bdo element
Client Image Map Module	Defines browser side image map elements
Edit Module	Defines the editing elements del and ins
Forms Module	Defines all elements used in forms
Frames Module	Defines the frameset elements
Hypertext Module	Defines the a element
Iframe Module	Defines the iframe element
Image Module	Defines the img element
Intrinsic Events Module	Defines event attributes like onblur and onchange
Legacy Module	Defines deprecated* elements and attributes
Link Module	Defines the link element
List Module	Defines the list elements ol, li, ul, dd, dt, and dl
Metainformation Module	Defines the meta element
Name Identification Module	Defines the deprecated* name attribute
Object Module	Defines the object and param elements
Presentation Module	Defines presentation elements like b and i
Scripting Module	Defines the script and noscript elements
Server Image Map Module	Defines server side image map elements
Structure Module	Defines the elements html, head, title and body
Style Attribute Module	Defines the style attribute
Style Sheet Module	Defines the style element
Tables Module	Defines the elements used in tables
Target Module	Defines the target attribute
Text Module	Defines text container elements like p and h1

* Deprecated elements should not be used in XHTML.

XHTML Standard Attributes

XHTML tags can have attributes. The special attributes for each tag are listed under each tag description. The attributes listed here are the core and language attributes that are standard for all tags (with a few exceptions).

Core Attributes

Not valid in base, head, html, meta, param, script, style, and title elements.

Attribute	Value	Description
class	*class_rule* or *style_rule*	The class of the element
id	*id_name*	A unique id for the element
style	*style_definition*	An inline style definition
title	*tooltip_text*	A text to display in a tool tip

Language Attributes

Not valid in base, br, frame, frameset, hr, iframe, param, and script elements.

Attribute	Value	Description
dir	ltr | rtl	Sets the text direction
lang	*language_code*	Sets the language code

Keyboard Attributes

Attribute	Value	Description
accesskey	*character*	Sets a keyboard shortcut to access an element
tabindex	*number*	Sets the tab order of an element

XHTML Event Attributes

New to HTML 4.0 was the ability to let HTML events trigger actions in the browser, like starting a JavaScript when a user clicks on an HTML element. Below is a list of attributes that can be inserted into HTML tags to define event actions.

If you want to learn more about programming with these events, you should study our JavaScript tutorial and our DHTML tutorial.

Window Events

Only valid in body and frameset elements

Attribute	Value	Description
onload	*script*	Script to be run when a document loads
onunload	*script*	Script to be run when a document unloads

Form Element Events

Only valid in form elements.

Attribute	Value	Description
onchange	*script*	Script to be run when the element changes
onsubmit	*script*	Script to be run when the form is submitted
onreset	*script*	Script to be run when the form is reset
onselect	*script*	Script to be run when the element is selected
onblur	*script*	Script to be run when the element loses focus
onfocus	*script*	Script to be run when the element gets focus

---

Keyboard Events

Not valid in base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, and title elements.

Attribute	Value	Description
onkeydown	*script*	What to do when key is pressed
onkeypress	*script*	What to do when key is pressed and released
onkeyup	*script*	What to do when key is released

Mouse Events

Not valid in base, bdo, br, frame, frameset, head, html, iframe, meta, param, script, style, and title elements.

Attribute	Value	Description
onclick	*script*	What to do on a mouse click
ondblclick	*script*	What to do on a mouse doubleclick
onmousedown	*script*	What to do when mouse button is pressed
onmousemove	*script*	What to do when mouse pointer moves
onmouseover	*script*	What to do when mouse pointer moves over an element
onmouseout	*script*	What to do when mouse pointer moves out of an element
onmouseup	*script*	What to do when mouse button is released

You Have Learned XHTML, Now What?

XHTML Summary

This tutorial has taught you how to create stricter and cleaner HTML pages.

You have learned that all XHTML elements must be properly nested, XHTML documents must be well-formed, all tag names must be in lowercase, and that all XHTML elements must be closed.

You have also learned that all XHTML documents must have a DOCTYPE declaration, and that the html, head, title, and body elements must be present.

For more information on XHTML, please look at our XHTML reference.

Now You Know XHTML, What's Next?

The next step is to learn CSS and JavaScript.

CSS

CSS is used to control the style and layout of multiple Web pages all at once.

With CSS, all formatting can be removed from the HTML document and stored in a separate file.

CSS gives you total control of the layout, without messing up the document content.

To learn how to create style sheets, please visit our CSS tutorial.

CSS Tutorial

CSS Tutorial

Save a lot of work with CSS!

In our CSS tutorial you will learn how to use CSS to control the style and layout of multiple Web pages all at once.

Start learning CSS now!

Introduction to CSS

What You Should Already Know

Before you continue you should have some basic understanding of the following:

- HTML / XHTML

If you want to study this subject first, find the tutorials on our Home page.

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles are normally stored in **Style Sheets**
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save you a lot of work
- External Style Sheets are stored in **CSS files**
- Multiple style definitions will **cascade** into one

CSS Demo

With CSS, your HTML documents can be displayed using different output styles:

Styles Solve a Common Problem

HTML tags were originally designed to define the content of a document. They were supposed to say "This is a header", "This is a paragraph", "This is a table", by using tags like <h1>, <p>, <table>, and so on. The layout of the document was supposed to be taken care of by the browser, without using any formatting tags.

As the two major browsers - Netscape and Internet Explorer - continued to add new HTML tags and attributes (like the <font> tag and the color attribute) to the original HTML specification, it became more and more difficult to create Web sites where the content of HTML documents was clearly separated from the document's presentation layout.

To solve this problem, the World Wide Web Consortium (W3C) - the non profit, standard setting consortium, responsible for standardizing HTML - created STYLES in addition to HTML 4.0.

All major browsers support Cascading Style Sheets.

Style Sheets Can Save a Lot of Work

Styles sheets define HOW HTML elements are to be displayed, just like the font tag and the color attribute in HTML 3.2. Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in your Web, just by editing one single CSS document!

CSS is a breakthrough in Web design because it allows developers to control the style and layout of multiple Web pages all at once. As a Web developer you can define a style for each HTML element and apply it to as many Web pages as you want. To make a global change, simply change the style, and all elements in the Web are updated automatically.

Multiple Styles Will Cascade Into One

Style sheets allow style information to be specified in many ways. Styles can be specified inside a single HTML element, inside the <head> element of an HTML page, or in an external CSS file. Even multiple external style sheets can be referenced inside a single HTML document.

Cascading Order

What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. Browser default
2. External style sheet
3. Internal style sheet (inside the <head> tag)
4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style declared inside the <head> tag, in an external style sheet, or in a browser (a default value).

CSS Syntax

Syntax

The CSS syntax is made up of three parts: a selector, a property and a value:

```
selector {property: value}
```

The selector is normally the HTML element/tag you wish to define, the property is the attribute you wish to change, and each property can take a value. The property and value are separated by a colon, and surrounded by curly braces:

```
body {color: black}
```

Note: If the value is multiple words, put quotes around the value:

```
p {font-family: "sans serif"}
```

Note: If you wish to specify more than one property, you must separate each property with a semicolon. The example below shows how to define a center aligned paragraph, with a red text color:

```
p {text-align:center;color:red}
```

To make the style definitions more readable, you can describe one property on each line, like this:

```
p
{
text-align: center;
color: black;
font-family: arial
}
```

Grouping

You can group selectors. Separate each selector with a comma. In the example below we have grouped all the header elements. All header elements will be displayed in green text color:

```
h1,h2,h3,h4,h5,h6
{
color: green
}
```

The class Selector

With the class selector you can define different styles for the same type of HTML element.

Say that you would like to have two types of paragraphs in your document: one right-aligned paragraph, and one center-aligned paragraph. Here is how you can do it with styles:

```
p.right {text-align: right}
p.center {text-align: center}
```

You have to use the class attribute in your HTML document:

```
<p class="right">
This paragraph will be right-aligned.
```

```
</p>
<p class="center">
This paragraph will be center-aligned.
</p>
```

Note: To apply more than one class per given element, the syntax is:

```
<p class="center bold">
This is a paragraph.
</p>
```

The paragraph above will be styled by the class "center" AND the class "bold".

You can also omit the tag name in the selector to define a style that will be used by all HTML elements that have a certain class. In the example below, all HTML elements with class="center" will be center-aligned:

```
.center {text-align: center}
```

In the code below both the h1 element and the p element have class="center". This means that both elements will follow the rules in the ".center" selector:

```
<h1 class="center">
This heading will be center-aligned
</h1>
<p class="center">
This paragraph will also be center-aligned.
</p>
```

Do **NOT** start a class name with a number! It will not work in Mozilla/Firefox.

Add Styles to Elements with Particular Attributes

You can also apply styles to HTML elements with particular attributes.

The style rule below will match all input elements that have a type attribute with a value of "text":

```
input[type="text"] {background-color: blue}
```

The id Selector

You can also define styles for HTML elements with the id selector. The id selector is defined as a #.

The style rule below will match the element that has an id attribute with a value of "green":

```
#green {color: green}
```

The style rule below will match the p element that has an id with a value of "para1":

```
p#para1
{
text-align: center;
color: red
}
```

Do **NOT** start an ID name with a number! It will not work in Mozilla/Firefox.

CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. A comment will be ignored by browsers. A CSS comment begins with "/*", and ends with "*/", like this:

```
/* This is a comment */
p
{
text-align: center;
/* This is another comment */
color: black;
font-family: arial
}
```

CSS How To...

Examples

- Look at **Example 1**

<html> <head>
<link rel=”stylesheet”
Type=”text/css” href=”ex1.css” />
</head>
<body> <h1> This header is 36 pt</h1>
<h2>This header is blue</h2>
This paragraph has a left margin of 50 pixels
</body> </html>

- Look at **Example 2**

<html> <head>
<link rel=”stylesheet”
Type=”text/css” href=”ex2.css” />
</head>
<body> <h1> This is a header 1</h1> <hr />
You can see that the stylesheet formats the text
This is a link
</body> </html>

How to Insert a Style Sheet

When a browser reads a style sheet, it will format the document according to it. There are three ways of inserting a style sheet:

External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

```
<head>
<link rel="stylesheet" type="text/css"
href="mystyle.css" />
</head>
```

The browser will read the style definitions from the file mystyle.css, and format the document according to it.

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension. An example of a style sheet file is shown below:

```
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
```

Do **NOT** leave spaces between the property value and the units! If you use "margin-left: 20 px" instead of "margin-left: 20px" it will only work properly in IE6 but it will not work in Mozilla/Firefox or Netscape.

Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section by using the <style> tag, like this:

```
<head>
<style type="text/css">
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
</style>
</head>
```

The browser will now read the style definitions, and format the document according to it.

Note: A browser normally ignores unknown tags. This means that an old browser that does not support styles, will ignore the <style> tag, but the content of the <style> tag will be displayed on the page. It is possible to prevent an old browser from displaying the content by hiding it in the HTML comment element:

```
<head>
<style type="text/css">
<!--
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
-->
</style>
</head>
```

Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly, such as when a style is to be applied to a single occurrence of an element.

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color: sienna; margin-left: 20px">
This is a paragraph
</p>
```

Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

For example, an external style sheet has these properties for the h3 selector:

```
h3
{
color: red;
text-align: left;
font-size: 8pt
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3
{
text-align: right;
font-size: 20pt
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
color: red;
text-align: right;
font-size: 20pt
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

CSS Background

The CSS background properties define the background effects of an element.

Examples

Set the background color
This example demonstrates how to set the background color for an element.

```
<html> <head>

<style type="text/css">

body {background-color: yellow}

h1 {background-color: #00ff00}

h2 {background-color: transparent}

p {background-color: rgb(250,0,255)}

</style> </head> <body>

<h1>This is header 1</h1>

<h2>This is header 2</h2>

<p>This is a paragraph</p>

</body> </html>
```

Set an image as the background
This example demonstrates how to set an image as the background.

```
<html><head><style type="text/css">body

{ background-image:url('bgdesert.jpg')}

</style></head><body></body></html>
```

How to repeat a background image
This example demonstrates how to repeat a background image.

```
<html><head><style type="text/css">body

{ background-image: url('bgdesert.jpg');background-repeat: repeat}</style>
```

</head><body></body></html>

How to repeat a background image only vertically
This example demonstrates how to repeat a background image only vertically.

<html><head><style type="text/css">body

{ background-image: url('bgdesert.jpg');background-repeat: repeat-y}

</style></head><body></body></html>

How to repeat a background image only horizontally
This example demonstrates how to repeat a background image only horizontally.

<html><head><style type="text/css">body

{ background-image: url('bgdesert.jpg');background-repeat: repeat-x}

</style></head><body></body></html>

How to display a background image only one time
This example demonstrates how to display a background image only one time

<html><head><style type="text/css">body

{ background-image: url('bgdesert.jpg');background-repeat: no-repeat}

</style></head><body></body></html>

How to place the background image
This example demonstrates how to place the image on the page.

<html><head><style type="text/css">body

{ background-image:url('smiley.gif');background-repeat:no-repeat;background-attachment:fixed;

background-position:center; }

</style></head><body>

Note: For this to work in Mozilla, the background-attachment property must be set to "fixed". </body></html>

How to position a background image using %
This example demonstrates how to position an image on the page using percent.

<html><head><style type="text/css">body

{ background-image: url('smiley.gif');background-repeat: no-repeat;

background-attachment:fixed;background-position: 30% 20%; }

</style></head><body>

Note: For this to work in Mozilla, the background-attachment property must be set to "fixed". </body></html>

How to position a background image using pixels

This example demonstrates how to position an image on the page using pixels.

```
<html><head><style type="text/css">body

{ background-image: url('smiley.gif');background-repeat: no-repeat;

background-attachment:fixed;background-position: 50px 100px;}

</style></head><body>

<p><b>Note:</b> For this to work in Mozilla, the background-attachment property must be set to "fixed".</p></body></html>
```

How to set a fixed background image

This example demonstrates how to set a fixed background image. The image will not scroll with the rest of the page.

```
<html><head><style type="text/css">body

{background-image: url('smiley.gif');background-repeat: no-repeat;background-attachment: fixed}

</style></head><body>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>

<p>The image will not scroll with the rest of the page</p>
```

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

The image will not scroll with the rest of the page

</body></html>

All the background properties in one declaration

This example demonstrates how to use the shorthand property for setting all of the background properties in one declaration.

<html><head><style type="text/css">body

{ background: #00ff00 url('smiley.gif') no-repeat fixed center; }</style></head><body>

This is some text This is some text This is some text

This is some text This is some text This is some text

This is some text This is some text This is some text

This is some text This is some text This is some text

This is some text This is some text This is some text

This is some text This is some text This is some text

This is some text This is some text This is some text

This is some text This is some text This is some text </body></html>

CSS Background Properties

The CSS background properties allow you to control the background color of an element, set an image as the background, repeat a background image vertically or horizontally, and position an image on a page.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
background	A shorthand property for setting all background properties in one declaration	*background-color* *background-image* *background-repeat* *background-attachment* *background-position*	4	1	6	1
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	scroll fixed	4	1	6	1
background-color	Sets the background color of an element	*color-rgb* *color-hex* *color-name* transparent	4	1	4	1
background-image	Sets an image as the background	url(*URL*) none	4	1	4	1
background-position	Sets the starting position of a background image	top left top center top right center left center center center right bottom left bottom center bottom right *x% y%* *xpos ypos*	4	1	6	1
background-repeat	Sets if/how a background image will be repeated	repeat repeat-x repeat-y no-repeat	4	1	4	1

CSS Text

The CSS text properties define the appearance of text.

Examples

Set the color of the text
This example demonstrates how to set the color of the text.

<html><head><style type="text/css"> h1 {color: #00ff00} h2 {color: #dda0dd}

p {color: rgb(0,0,255)}</style></head>

<body><h1>This is header 1</h1><h2>This is header 2</h2> This is a paragraph

</body></html>

Set the background-color of the text
This example demonstrates how to set the background-color of a part of the text.

<html><head><style type="text/css">span.highlight

{background-color:yellow}</style></head>

<body>

This is a text. This is a text. This is a text. This is a text. This is a text. This is a text. This is a text. This is a text. This is a text. This is a text. </body></html>

Specify the space between characters
This example demonstrates how to increase or decrease the space between characters.

<html><head><style type="text/css">h1 {letter-spacing: -3px}h4 {letter-spacing: 0.5cm}

</style></head><body><h1>This is header 1</h1><h4>This is header 4</h4></body>

</html>

Specify the space between lines
This example demonstrates how to specify the space between the lines in a paragraph.<html>

<head><style type="text/css">p.small {line-height: 90%}p.big {line-height: 200%}</style>

</head><body> This is a paragraph with a standard line-height.

The default line height in most browsers is about 110% to 120%.

This is a paragraph with a standard line-height.This is a paragraph with a standard line-height.

This is a paragraph with a smaller line-height.This is a paragraph with a smaller line-height.

This is a paragraph with a smaller line-height.This is a paragraph with a smaller line-height.

This is a paragraph with a bigger line-height.This is a paragraph with a bigger line-height.

This is a paragraph with a bigger line-height.This is a paragraph with a bigger line-height.

</body></html>

Align the text

This example demonstrates how to align the text.

```
<html><head><style type="text/css">h1 {text-align: center}h2 {text-align: left}

h3 {text-align: right}</style></head>

<body><h1>This is header 1</h1><h2>This is header 2</h2><h3>This is header 3</h3></body>

</html>
```

Decorate the text

This example demonstrates how to add decoration to text.

```
<html><head><style type="text/css">h1 {text-decoration: overline}

h2 {text-decoration: line-through}h3 {text-decoration: underline}a {text-decoration: none}</style>

</head>

<body><h1>This is header 1</h1><h2>This is header 2</h2><h3>This is header 3</h3>

<p><a href="http://www.w3schools.com/default.asp">This is a link</a></p></body></html>
```

Indent text

This example demonstrates how to indent the first line of a paragraph.

```
<html><head><style type="text/css">p {text-indent: 1cm}</style></head>

<body><p>This is some text in a paragraph This is some text in a paragraph

This is some text in a paragraph This is some text in a paragraph This is some text in a paragraph

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

Control the letters in a text

This example demonstrates how to control the letters in a text.

```
<html><head><style type="text/css">p.uppercase {text-transform: uppercase}

p.lowercase {text-transform: lowercase} p.capitalize {text-transform: capitalize}</style></head>

<body><p class="uppercase">This is some text in a paragraph</p>

<p class="lowercase">This is some text in a paragraph</p>

<p class="capitalize">This is some text in a paragraph</p></body></html>
```

Set the text direction of an element

This example demonstrates how to change the text direction of an element.

```
<html><head><style type="text/css">div.one
```

```
{direction: rtl}  div.two

{direction: ltr}  </style></head>

<body>

<div class="one">Some text. Right-to-left direction.</div>

<div class="two">Some text. Left-to-right direction.</div></body></html>
```

Increase the white space between words

This example demonstrates how to increase the white space between words in a paragraph.

```
<html><head><style type="text/css">p

{ word-spacing: 30px}</style></head>

<body>

<p>This is some text. This is some text.</p>

</body></html>
```

Disable text wrapping inside an element

This example demonstrates how to disable text wrapping inside an element.

```
<html><head><style type="text/css">p

{white-space: nowrap}

</style></head><body>

<p>This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.</p></body></html>
```

CSS Text Properties

The CSS text properties allow you to control the appearance of text. It is possible to change the color of a text, increase or decrease the space between characters in a text, align a text, decorate a text, indent the first line in a text, and more.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
color	Sets the color of a text	*color*	3	1	4	1
direction	Sets the text direction	ltr rtl	6	1	6	2
line-height	Sets the distance between lines	normal *number* *length* *%*	4	1	4	1
letter-spacing	Increase or decrease the space between characters	normal *length*	4	1	6	1
text-align	Aligns the text in an element	left right center justify	4	1	4	1
text-decoration	Adds decoration to text	none underline overline line-through blink	4	1	4	1
text-indent	Indents the first line of text in an element	*length* *%*	4	1	4	1
text-shadow		none *color* *length*				
text-transform	Controls the letters in an element	none capitalize uppercase lowercase	4	1	4	1
unicode-bidi		normal embed bidi-override	5			2
white-space	Sets how white space inside an element is handled	normal pre nowrap	5	1	4	1
word-spacing	Increase or decrease the space between words	normal *length*	6	1	6	1

CSS Font

The CSS font properties define the font in text.

Examples

Set the font of a text
This example demonstrates how to set a font of a text.

<html><head><style type="text/css">h3 {font-family: times}p {font-family: courier}

p.sansserif {font-family: sans-serif}</style></head>

<body><h3>This is header 3</h3> This is a paragraph

This is a paragraph </body></html>

Set a paragraph font using the "caption" value
This example demonstrates how to set a paragraph font using the "caption" value.

<html><body> This is a normal paragraph

This is a paragraph with a "caption" font </body></html>

Set the size of the font
This example demonstrates how to set the size of a font.

<html><head><style type="text/css">h1 {font-size: 150%}h2 {font-size: 130%}

p {font-size: 100%}</style></head>

<body><h1>This is header 1</h1><h2>This is header 2</h2> This is a paragraph </body>

</html>

Set the size of the font using font-size-adjust
This example demonstrates how to set the size of a font using font-size-adjust.

<html><head><style type="text/css">h1 {font-size-adjust: 0.50}h2 {font-size-adjust: 0.40}

p {font-size-adjust: 0.60}</style></head><body><h1>This is header 1</h1>

<h2>This is header 2</h2> This is a paragraph </body></html>

Set the style of the font
This example demonstrates how to set the style of a font.

<html><head><style type="text/css">h1 {font-style: italic}h2 {font-style: normal}

p {font-style: oblique}</style></head>

<body><h1>This is header 1</h1><h2>This is header 2</h2> This is a paragraph </body>

</html>

Set the variant of the font
This example demonstrates how to set the variant of a font.

<html><head><style type="text/css">p.normal {font-variant: normal}

p.small {font-variant: small-caps}</style></head>

<body> This is a paragraph This is a paragraph

</body></html>

Set the boldness of the font

This example demonstrates how to set the boldness of a font.

```
<html><head><style type="text/css">p.normal {font-weight: normal}

p.thick {font-weight: bold}p.thicker {font-weight: 900}</style></head>

<body><p class="normal">This is a paragraph</p>

<p class="thick">This is a paragraph</p><p class="thicker">This is a paragraph</p></body>

</html>
```

All the font properties in one declaration

This example demonstrates how to use the shorthand property for setting all of the font properties in one declaration.

```
<html><head><style type="text/css">p

{font: italic small-caps 900 12px arial}</style></head>

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

CSS Font Properties

The CSS font properties allow you to change the font family, boldness, size, and the style of a text.

Note: In CSS1 fonts are identified by a font name. If a browser does not support the specified font, it will use a default font.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
font	A shorthand property for setting all of the properties for a font in one declaration	*font-style* *font-variant* *font-weight* *font-size/line-height* *font-family* caption icon menu message-box small-caption status-bar	4	1	4	1
font-family	A prioritized list of font family names and/or generic family names for an element	*family-name* *generic-family*	3	1	4	1
font-size	Sets the size of a font	xx-small	3	1	4	1

		x-small small medium large x-large xx-large smaller larger *length* *%*				
font-size-adjust	Specifies an aspect value for an element that will preserve the x-height of the first-choice font	none *number*	-	-	-	2
font-stretch	Condenses or expands the current font-family	normal wider narrower ultra-condensed extra-condensed condensed semi-condensed semi-expanded expanded extra-expanded ultra-expanded	-	-	-	2
font-style	Sets the style of the font	normal italic oblique	4	1	4	1
font-variant	Displays text in a small-caps font or a normal font	normal small-caps	4	1	6	1
font-weight	Sets the weight of a font	normal bold bolder lighter 100 200 300 400 500 600 700 800 900	4	1	4	1

CSS Border

The CSS border properties define the borders around an element.

Examples

All the border properties in one declaration

This example demonstrates a shorthand property for setting all of the properties for the four borders in one declaration, can have from one to three values.

```
<html><head><style type="text/css">p

{border: medium double rgb(250,0,255)}</style></head>

<body><p>Some text</p></body></html>
```

Set different borders on each side

This example demonstrates how to set different borders on each side of the element.

```
<html><head><style type="text/css">p.soliddouble {border-style: solid double}

p.doublesolid {border-style: double solid}p.groovedouble {border-style: groove double}

p.three {border-style: solid double groove}</style></head>

<body><p class="soliddouble">Some text</p><p class="doublesolid">Some text</p>

<p class="groovedouble">Some text</p><p class="three">Some text</p></body></html>
```

All the top border properties in one declaration

This example demonstrates a shorthand property for setting all of the properties for the top border in one declaration.

```
<html><head><style type="text/css">p

{border-top: medium solid #ff0000}</style></head>

<body><p>Some text.</p></body></html>
```

All the bottom border properties in one declaration

This example demonstrates a shorthand property for setting all of the properties for the bottom border in one declaration.

```
<html><head><style type="text/css">p

{border-bottom: medium solid #ff0000}</style></head>

<body><p>Some text.</p></body></html>
```

All the left border properties in one declaration

This example demonstrates a shorthand property for setting all of the properties for the left border in one declaration.

```
<html><head><style type="text/css">p

{border-left: medium solid #ff0000}</style></head>

<body><p>Some text.</p></body></html>
```

All the right border properties in one declaration

This example demonstrates a shorthand property for setting all of the properties for the right border in one declaration.

```
<html><head><style type="text/css">p

{border-right: medium solid #ff0000}</style></head>

<body><p>Some text.</p></body></html>
```

Set the style of the four borders

This example demonstrates how to set the style of the four borders.

```
<html><head><style type="text/css">p.dotted {border-style: dotted}

p.dashed {border-style: dashed} p.solid {border-style: solid} p.double {border-style: double}

p.groove {border-style: groove} p.ridge {border-style: ridge} p.inset {border-style: inset}

p.outset {border-style: outset} </style></head>

<body> <p class="dotted">A dotted border</p> <p class="dashed">A dashed border</p>

<p class="solid">A solid border</p> <p class="double">A double border</p>

<p class="groove">A groove border</p> <p class="ridge">A ridge border</p>

<p class="inset">An inset border</p> <p class="outset">An outset border</p> </body></html>
```

Set the style of the top border

This example demonstrates how to set the style of the top border.

```
<html><head><style type="text/css">p.dotted {border-top-style: dotted}

p.dashed {border-top-style: dashed} p.solid {border-top-style: solid}

p.double {border-top-style: double} p.groove {border-top-style: groove}

p.ridge {border-top-style: ridge} p.inset {border-top-style: inset}

p.outset {border-top-style: outset} </style></head>

<body><p class="dotted">A dotted border</p> <p class="dashed">A dashed border</p>

<p class="solid">A solid border</p> <p class="double">A double border</p>

<p class="groove">A groove border</p> <p class="ridge">A ridge border</p>

<p class="inset">An inset border</p> <p class="outset">An outset border</p></body></html>
```

Set the style of the bottom border

This example demonstrates how to set the style of the bottom border.

```
<html><head><style type="text/css"> p.dotted {border-bottom-style: dotted}

p.dashed {border-bottom-style: dashed} p.solid {border-bottom-style: solid}
```

p.double {border-bottom-style: double} p.groove {border-bottom-style: groove}

p.ridge {border-bottom-style: ridge} p.inset {border-bottom-style: inset}

p.outset {border-bottom-style: outset} </style></head>

<body> A dotted border A dashed border

A solid border A double border

A groove border A ridge border

An inset border An outset border </body></html>

Set the style of the left border

This example demonstrates how to set the style of the left border.

<html><head><style type="text/css">p.dotted {border-left-style: dotted}

p.dashed {border-left-style: dashed} p.solid {border-left-style: solid}

p.double {border-left-style: double} p.groove {border-left-style: groove}

p.ridge {border-left-style: ridge} p.inset {border-left-style: inset}

p.outset {border-left-style: outset} </style></head>

<body> A dotted border A dashed border

A solid border A double border

A groove border A ridge border

An inset border An outset border </body></html>

Set the style of the right border

This example demonstrates how to set the style of the right border.

<html><head><style type="text/css"> p.dotted {border-right-style: dotted}

p.dashed {border-right-style: dashed} p.solid {border-right-style: solid}

p.double {border-right-style: double} p.groove {border-right-style: groove}

p.ridge {border-right-style: ridge} p.inset {border-right-style: inset}

p.outset {border-right-style: outset} </style> </head>

<body> A dotted border A dashed border

A solid border A double border

A groove border A ridge border

An inset border An outset border </body></html>

All the width of the border properties in one declaration
This example demonstrates a shorthand property for setting the width of the four borders in one declaration, can have from one to four values.

```
<html><head><style type="text/css">p.one

{border-style: solid;border-width: 5px}

p.two

{border-style: solid;border-width: thick}

p.three

{border-style: solid;border-width: 5px 10px}

p.four

{border-style: solid;border-width: 5px 10px 1px}

p.five

{border-style: solid;border-width: 5px 10px 1px medium}</style></head>

<body><p class="one">Some text</p> <p class="two">Some text</p>

<p class="three">Some text</p> <p class="four">Some text</p>

<p class="five">Some text</p>

<p><b>Note:</b> The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p></body></html>
```

Set the width of the top border
This example demonstrates how to set the width of the top border.

```
<html><head><style type="text/css">p.one

{border-style: solid;border-top-width: 15px}

p.two

{border-style: solid;border-top-width: thin}</style></head><body>

<p class="one"><b>Note:</b> The "border-top-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p>

<p class="two">Some text. Some more text.</p></body></html>
```

Set the width of the bottom border
This example demonstrates how to set the width of the bottom border.

```
<html><head><style type="text/css">p.one

{border-style: solid;border-bottom-width: 15px}

p.two

{border-style: solid;border-bottom-width: thin}</style></head><body>

<p class="one"><b>Note:</b> The "border-bottom-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p>

<p class="two">Some text. Some more text.</p></body></html>
```

Set the width of the left border

This example demonstrates how to set the width of the left border.

```
<html><head><style type="text/css">p.one

{border-style: solid;border-left-width: 15px}

p.two

{border-style: solid;border-left-width: thin}

</style></head><body>

<p class="one"><b>Note:</b> The "border-left-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p>

<p class="two">Some text. Some more text.</p></body></html>
```

Set the width of the right border

This example demonstrates how to set the width of the right border.

```
<html><head><style type="text/css">p.one

{border-style: solid;border-right-width: 15px}

p.two

{border-style: solid;border-right-width: thin}

</style></head><body>

<p class="one"><b>Note:</b> The "border-right-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p>

<p class="two">Some text. Some more text.</p></body></html>
```

Set the color of the four borders

This example demonstrates how to set the color of the four borders. It can have from one to four colors.

```
<html><head><style type="text/css">p.one

{border-style: solid;border-color: #0000ff}

p.two

{border-style: solid;border-color: #ff0000 #0000ff}

p.three

{border-style: solid;border-color: #ff0000 #00ff00 #0000ff}

p.four

{border-style: solid;border-color: #ff0000 #00ff00 #0000ff rgb(250,0,255)}</style></head>

<body><p class="one">One-colored border!</p> <p class="two">Two-colored border!</p>

<p class="three">Three-colored border!</p> <p class="four">Four-colored border!</p>

<p><b>Note:</b> The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p></body></html>
```

Set the color of the top border
This example demonstrates how to set the color of the top border.

```
<html><head><style type="text/css">p

{border-style: solid;border-top-color: #ff0000}</style></head>

<body><p>Some text.</p></body></html>
```

Set the color of the bottom border
This example demonstrates how to set the color of the bottom border.

```
<html><head><style type="text/css">p

{border-style: solid;border-bottom-color: #ff0000}</style></head>

<body><p>Some text.</p></body></html>
```

Set the color of the left border
This example demonstrates how to set the color of the left border.

```
<html><head><style type="text/css">p

{border-style: solid;border-left-color: #ff0000}</style></head>

<body><p>Some text.</p></body></html>
```

Set the color of the right border
This example demonstrates how to set the color of the right border.

<html><head><style type="text/css">p

{border-style: solid;border-right-color: #ff0000}</style></head>

<body> Some text. </body></html>

CSS Border Properties

The CSS border properties allow you to specify the style and color of an element's border. In HTML we use tables to create borders around a text, but with the CSS border properties we can create borders with nice effects, and it can be applied to any element.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
border	A shorthand property for setting all of the properties for the four borders in one declaration	*border-width* *border-style* *border-color*	4	1	4	1
border-bottom	A shorthand property for setting all of the properties for the bottom border in one declaration	*border-bottom-width* *border-style* *border-color*	4	1	6	1
border-bottom-color	Sets the color of the bottom border	*border-color*	4	1	6	2
border-bottom-style	Sets the style of the bottom border	*border-style*	4	1	6	2
border-bottom-width	Sets the width of the bottom border	thin medium thick *length*	4	1	4	1
border-color	Sets the color of the four borders, can have from one to four colors	*color*	4	1	6	1
border-left	A shorthand property for setting all of the properties for the left border in one declaration	*border-left-width* *border-style* *border-color*	4	1	6	1
border-left-color	Sets the color of the left border	*border-color*	4	1	6	2
border-left-style	Sets the style of the left border	*border-style*	4	1	6	2
border-left-width	Sets the width of the left border	thin medium thick *length*	4	1	4	1
border-right	A shorthand property for	*border-right-width*	4	1	6	1

	setting all of the properties for the right border in one declaration	*border-style* *border-color*				
border-right-color	Sets the color of the right border	*border-color*	4	1	6	2
border-right-style	Sets the style of the right border	*border-style*	4	1	6	2
border-right-width	Sets the width of the right border	thin medium thick *length*	4	1	4	1
border-style	Sets the style of the four borders, can have from one to four styles	none hidden dotted dashed solid double groove ridge inset outset	4	1	6	1
border-top	A shorthand property for setting all of the properties for the top border in one declaration	*border-top-width* *border-style* *border-color*	4	1	6	1
border-top-color	Sets the color of the top border	*border-color*	4	1	6	2
border-top-style	Sets the style of the top border	*border-style*	4	1	6	2
border-top-width	Sets the width of the top border	thin medium thick *length*	4	1	4	1
border-width	A shorthand property for setting the width of the four borders in one declaration, can have from one to four values	thin medium thick *length*	4	1	4	1

CSS Outlines

The CSS outline properties is used to draw a line around an element, outside the border edge.

Examples

Draw a line around an element (outline) (does not work in IE)
This example demonstrates how to draw a line around an element, outside the border edge.

<html><head><style type="text/css">p

{border: red solid thin;outline: green dotted thick}</style></head><body>

Some text. </body></html>

Set the style of an outline (does not work in IE)
This example demonstrates how to set the style of an outline.

<html><head><style type="text/css">p

{border: red solid thin;} p.dotted {outline-style: dotted} p.dashed {outline-style: dashed}

p.solid {outline-style: solid} p.double {outline-style: double} p.groove {outline-style: groove}

p.ridge {outline-style: ridge} p.inset {outline-style: inset} p.outset {outline-style: outset}</style>

</head>

<body> A dotted outline A dashed outline

A solid outline A double outline

A groove outline A ridge outline

An inset outline An outset outline </body></html>

Set the color of an outline (does not work in IE)
This example demonstrates how to set the color of an outline.

<html><head><style type="text/css">p

{border: red solid thin;outline-style: solid;outline-color: #00ff00}</style></head><body>

Some text. </body></html>

Set the width of an outline (does not work in IE)
This example demonstrates how to set the width of an outline.

<html><head><style type="text/css">p.one

{border: red solid thin;outline-style: solid;outline-width: thick}

p.two

{border: red solid thin;outline-style: solid;outline-width: 2px}</style></head>

<body> Some text. Some text. </body></html>

CSS Outline Properties

An outline is a line that is drawn around elements, outside the border edge, to make the element "stand out".

The CSS outline properties sets the outlines around elements. You can specify the style, color, and width of the outline.

Note: Outlines do not take up space, and they do not have to be rectangular.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
outline	A shorthand property for setting all the outline properties in one declaration	*outline-color* *outline-style* *outline-width*	-	1.5	-	2
outline-color	Sets the color of the outline around an element	*color* invert	-	1.5	-	2
outline-style	Sets the style of the outline around an element	none dotted dashed solid double groove ridge inset outset	-	1.5	-	2
outline-width	Sets the width of the outline around an element	thin medium thick *length*	-	1.5	-	2

CSS Margin

The CSS margin properties define the space around elements.

Examples

All the margin properties in one declaration
This example demonstrates how to set a shorthand property for setting all of the margin properties in one declaration.

<html><head><style type="text/css">p.margin {margin: 2cm 4cm 3cm 4cm}</style></head>

<body> This is a paragraph with no specified margins

```
<p class="margin">This is a paragraph with specified margins</p>

<p>This is a paragraph with no specified margins</p></body></html>
```

Set the top margin of a text using a cm value
This example demonstrates how to set the top margin of a text using a cm value.

```
<html><head><style type="text/css">p.topmargin {margin-top: 5cm}</style></head>

<body><p>This is a paragraph with no margin specified</p>

<p class="topmargin">This is a paragraph with a specified top margin</p></body></html>
```

Set the top margin of a text using a percent value
This example demonstrates how to set the top margin of a text using a percent value.

```
<html><head><style type="text/css">

p.topmargin

{margin-top: 25%}</style></head><body>

<p>This is a paragraph with no margin specified</p>

<p class="topmargin">This is a paragraph with a specified top margin</p></body></html>
```

Set the bottom margin of a text using a cm value
This example demonstrates how to set the bottom margin of a text using a cm value.

```
<html><head><style type="text/css">p.bottommargin {margin-bottom: 2cm}</style></head>

<body><p>This is a paragraph with no margin specified</p>

<p class="bottommargin">This is a paragraph with a specified bottom margin</p>

<p>This is a paragraph with no margin specified</p></body></html>
```

Set the bottom margin of a text using a percent value
This example demonstrates how to set the bottom margin of a text using a percent value.

```
<html><head><style type="text/css">

p.bottommargin

{margin-bottom: 25%}</style></head>

<body><p>This is a paragraph with no margin specified</p>

<p class="bottommargin">This is a paragraph with a specified bottom margin</p>

<p>This is a paragraph with no margin specified</p></body></html>
```

Set the left margin of a text using a cm value
This example demonstrates how to set the left margin of a text using a cm value.

```
<html><head><style type="text/css">p.leftmargin {margin-left: 2cm}</style></head>

<body><p>This is a paragraph with no margin specified</p>

<p class="leftmargin">This is a paragraph with a specified left margin</p></body></html>
```

Set the left margin of a text using a percent value
This example demonstrates how to set the left margin of a text using a percent value.

```
<html><head><style type="text/css">

p.leftmargin

{margin-left: 25%}</style></head>

<body><p>This is a paragraph with no margin specified</p>

<p class="leftmargin">This is a paragraph with a specified left margin</p></body></html>
```

Set the right margin of a text using a cm value
This example demonstrates how to set the right margin of a text using a cm value.

```
<html><head><style type="text/css">p.rightmargin {margin-right: 5cm}</style></head>

<body>

<p style="text-align:right">This is a right aligned paragraph with no margin specified</p>

<p class="rightmargin" style="text-align:right">This is a right aligned paragraph with a specified right
margin</p></body></html>
```

Set the right margin of a text using a percent value
This example demonstrates how to set the right margin of a text using a percent value.

```
<html><head><style type="text/css">

p.rightmargin

{margin-right:25%}</style></head>

<body><p style="text-align:right">This is a right aligned paragraph with no margin specified</p>

<p class="rightmargin" style="text-align:right">This is a right aligned paragraph with a specified right
margin</p></body></html>
```

CSS Margin Properties

The CSS margin properties define the space around elements. It is possible to use negative values to overlap content. The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used to change all of the margins at once.

Note: Netscape and IE give the body tag a default margin of 8px. Opera does not! Instead, Opera applies a default padding of 8px, so if one wants to adjust the margin for an entire page and have it display correctly in Opera, the body padding must be set as well!

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
margin	A shorthand property for setting the margin properties in one declaration	*margin-top* *margin-right* *margin-bottom* *margin-left*	4	1	4	1
margin-bottom	Sets the bottom margin of an element	auto *length* *%*	4	1	4	1
margin-left	Sets the left margin of an element	auto *length* *%*	3	1	4	1
margin-right	Sets the right margin of an element	auto *length* *%*	3	1	4	1
margin-top	Sets the top margin of an element	auto *length* *%*	3	1	4	1

CSS Padding

The CSS padding properties define the space between the element border and the element content.

Examples

All the padding properties in one declaration
This example demonstrates a shorthand property for setting all of the padding properties in one declaration, can have from one to four values.

<html><head><style type="text/css">td.test1 {padding: 1.5cm}td.test2 {padding: 0.5cm 2.5cm}

</style></head>

```
<body><table border="1"><tr><td class="test1">

This is a tablecell with equal padding on each side.

</td></tr></table><br /><table border="1"><tr><td class="test2">

This tablecell has a top and bottom padding of 0.5cm and a left and right padding of 2.5cm.

</td></tr></table></body></html>
```

Set the top padding using a cm value
This example demonstrates how to set the top padding of a table cell using a cm value.

```
<html><head><style type="text/css">td {padding-top: 2cm}</style></head>

<body><table border="1"><tr><td>This is a tablecell with a top padding</td></tr>

</table></body></html>
```

Set the top padding using a percent value
This example demonstrates how to set the top padding of a table cell using a percent value.

```
<html><head><style type="text/css">

td

{padding-top: 10%}</style></head>

<body><table border="1"><tr><td>This is a tablecell with a top padding</td></tr>

</table></body></html>
```

Set the bottom padding using a cm value
This example demonstrates how to set the bottom padding of a table cell using a cm value.

```
<html><head><style type="text/css">td {padding-bottom: 2cm}</style></head>

<body><table border="1"><tr><td>This is a tablecell with a bottom padding</td></tr>

</table></body></html>
```

Set the bottom padding using a percent value
This example demonstrates how to set the bottom padding of a table cell using a percent value.

```
<html><head><style type="text/css">td

{padding-bottom: 10%}</style></head>

<body><table border="1"><tr><td>This is a tablecell with a bottom padding</td></tr>

</table></body></html>
```

Set the left padding using a cm value
This example demonstrates how to set the left padding of a table cell using a cm value.

```
<html><head><style type="text/css">td {padding-left: 2cm}</style></head>

<body><table border="1"><tr><td>

This is a tablecell with a left padding. This is a tablecell with a left padding.</td></tr></table>

</body></html>
```

Set the left padding using a percent value
This example demonstrates how to set the left padding of a table cell using a percent value.

```
<html><head><style type="text/css">td

{padding-left: 10%}</style></head>

<body><table border="1"><tr><td>This is a tablecell with a left padding</td></tr>

</table></body></html>
```

Set the right padding using a cm value
This example demonstrates how to set the right padding of a table cell using a cm value.

```
<html><head><style type="text/css">td {padding-right: 5cm}</style></head>

<body><table border="1"><tr><td>This is a tablecell with a right padding. This is a tablecell with a right padding.</td></tr></table></body></html>
```

Set the right padding using a percent value
This example demonstrates how to set the right padding of a table cell using a percent value.

```
<html><head><style type="text/css">td{padding-right: 10%}</style></head>

<body><table border="1"><tr><td>This is a tablecell with a right padding</td></tr>

</table></body></html>
```

CSS Padding Properties

The CSS padding properties define the space between the element border and the element content. Negative values are not allowed. The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property is also created to control multiple sides at once.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
padding	A shorthand property for setting all of the padding properties in one declaration	*padding-top* *padding-right* *padding-bottom* *padding-left*	4	1	4	1
padding-bottom	Sets the bottom padding of an element	*length* *%*	4	1	4	1
padding-left	Sets the left padding of an element	*length* *%*	4	1	4	1
padding-right	Sets the right padding of an element	*length* *%*	4	1	4	1
padding-top	Sets the top padding of an element	*length* *%*	4	1	4	1

CSS List

The CSS list properties allow you to place the list-item marker, change between different list-item markers, or set an image as the list-item marker.

Examples

The different list-item markers in unordered lists
This example demonstrates the different list-item markers in CSS.

<html><head><style type="text/css"> ul.disc {list-style-type: disc} ul.circle {list-style-type: circle}

ul.square {list-style-type: square} ul.none {list-style-type: none}</style></head>

<body><ul class="disc"> Coffee Tea Coca Cola

<ul class="circle"> Coffee Tea Coca Cola

<ul class="square"> Coffee Tea Coca Cola

<ul class="none"> Coffee Tea Coca Cola </body> </html>

The different list-item markers in ordered lists
This example demonstrates the different list-item markers in CSS.

<html><head>

<style type="text/css"> ol.decimal {list-style-type: decimal} ol.lroman {list-style-type: lower-roman}

ol.uroman {list-style-type: upper-roman} ol.lalpha {list-style-type: lower-alpha}

ol.ualpha {list-style-type: upper-alpha}</style></head>

<body><ol class="decimal"> Coffee Tea Coca Cola

<ol class="lroman"> Coffee Tea Coca Cola

<ol class="uroman"> Coffee Tea Coca Cola

<ol class="lalpha"> Coffee Tea Coca Cola

<ol class="ualpha"> Coffee Tea Coca Cola </body></html>

All the list style types

This example demonstrates all the different list-item markers in CSS.

<html><head><style type="text/css"> ul.disc {list-style-type: disc}

ul.circle {list-style-type: circle} ul.square {list-style-type: square} ul.none {list-style-type: none}

ul.decimal {list-style-type: decimal} ul.decimal-leading-zero {list-style-type: decimal-leading-zero}

ul.lower-roman {list-style-type: lower-roman} ul.upper-roman {list-style-type: upper-roman}

ul.lower-alpha {list-style-type: lower-alpha} ul.upper-alpha {list-style-type: upper-alpha}

ul.lower-greek {list-style-type: lower-greek} ul.lower-latin {list-style-type: lower-latin}

ul.upper-latin {list-style-type: upper-latin} ul.hebrew {list-style-type: hebrew}

ul.armenian {list-style-type: armenian} ul.georgian {list-style-type: georgian}

ul.cjk-ideographic {list-style-type: cjk-ideographic} ul.hiragana {list-style-type: hiragana}

ul.katakana {list-style-type: katakana} ul.hiragana-iroha {list-style-type: hiragana-iroha}

ul.katakana-iroha {list-style-type: katakana-iroha} </style> </head>

<body><ul class="disc"> Disc type Tea Coca Cola

<ul class="circle"> Circle type Tea Coca Cola

<ul class="square"> Square type Tea Coca Cola

<ul class="none"> The "none" type Tea Coca Cola

<ul class="decimal"> Decimal type Tea Coca Cola

<ul class="decimal-leading-zero"> Decimal-leading-zero type Tea

Coca Cola

<ul class="lower-roman"> Lower-roman type Tea Coca Cola

<ul class="upper-roman"> Upper-roman type Tea Coca Cola

<ul class="lower-alpha"> Lower-alpha type Tea Coca Cola

<ul class="upper-alpha"> Upper-alpha type Tea Coca Cola

<ul class="lower-greek"> Lower-greek type Tea Coca Cola

<ul class="lower-latin"> Lower-latin type Tea Coca Cola

<ul class="upper-latin"> Upper-latin type Tea Coca Cola

<ul class="hebrew"> Hebrew type Tea Coca Cola

<ul class="armenian"> Armenian type Tea Coca Cola

<ul class="georgian"> Georgian type Tea Coca Cola

<ul class="cjk-ideographic"> Cjk-ideographic type Tea Coca Cola

<ul class="hiragana"> Hiragana type Tea Coca Cola

<ul class="katakana"> Katakana type Tea Coca Cola

<ul class="hiragana-iroha"> Hiragana-iroha type Tea Coca Cola

<ul class="katakana-iroha"> Katakana-iroha type Tea Coca Cola

</body></html>

Set an image as the list-item marker

This example demonstrates how to set an image as the list-item marker.

<html><head><style type="text/css">ul

{list-style-image: url('arrow.gif')}</style></head>

<body>
Coffee Tea Coca Cola </body> </html>

Place the list-item marker

This example demonstrates where to place the list-item marker.

<html><head> <style type="text/css">

ul.inside {list-style-position: inside}

ul.outside { list-style-position: outside } </style> </head>

<body> This list has a list-style-position with a value of "inside":

<ul class="inside"> Earl Grey Tea - A fine black tea

Jasmine Tea - A fabulous "all purpose" tea

Honeybush Tea - A super fruity delight tea

This list has a list-style-position with a value of "outside": <ul class="outside">

Earl Grey Tea - A fine black tea
Jasmine Tea - A fabulous "all purpose" tea

Honeybush Tea - A super fruity delight tea </body> </html>

All list properties in one declaration
This example demonstrates a shorthand property for setting all of the properties for a list in one declaration.

<html> <head> <style type="text/css">ul {list-style: square inside url('arrow.gif')}</style></head>

<body> Coffee Tea Coca Cola </body> </html>

CSS List Properties : The CSS list properties allow you to place the list-item marker, change between different list-item markers, or set an image as the list-item marker.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
list-style	A shorthand property for setting all of the properties for a list in one declaration	*list-style-type* *list-style-position* *list-style-image*	4	1	6	1
list-style-image	Sets an image as the list-item marker	none *url*	4	1	6	1
list-style-position	Sets where the list-item marker is placed in the list	inside outside	4	1	6	1
list-style-type	Sets the type of the list-item marker	none disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-alpha upper-alpha lower-greek lower-latin upper-latin hebrew armenian georgian cjk-ideographic hiragana katakana hiragana-iroha katakana-iroha	4	1	4	1
marker-offset		auto *length*		1	7	2

CSS Table

The CSS table properties allow you to set the layout of a table.

Examples

Set the layout of a table
This example demonstrates how to set the layout of a table.

<html> <head> <style type="text/css"> table.one

{table-layout: automatic}

table.two

{table-layout: fixed}</style></head>

<body> <table class="one" border="1" width="100%"><tr>

<td width="20%">10000000000000000000000000000</td> <td width="40%">10000000</td>

<td width="40%">100</td> </tr> </table>

<table class="two" border="1" width="100%"> <tr>

<td width="20%">10000000000000000000000000000</td> <td width="40%">10000000</td>

<td width="40%">100</td> </tr> </table> </body> </html>

Show empty cells in a table
This example demonstrates whether or not to show empty cells in a table.

<html><head> <style type="text/css">

Table { border-collapse: separate; empty-cells: show } </style> </head>

<body><table border="1"> <tr> <td>Peter</td> <td>Griffin</td> </tr> <tr> <td>Lois</td>

<td></td> </tr> </table> </body> </html>

Collapse a table border
This example demonstrates whether the table borders are collapsed into a single border or detached as in standard HTML.

<html><head> <style type="text/css">

table.coll { border-collapse: collapse }

table.sep { border-collapse: separate } </style> </head>

<body> <table class="coll" border="1"> <tr> <td>Peter</td> <td>Griffin</td> </tr>

<tr> <td>Lois</td> <td>Griffin</td> </tr> </table>

 <table class="sep" border="1"> <tr> <td>Peter</td> <td>Griffin</td> </tr> <tr>

<td>Lois</td> <td>Griffin</td> </tr> </table> </body> </html>

Set the space between table borders
This example demonstrates how to set the distance between cell borders.

<html> <head> <style type="text/css">

table.one { border-collapse: separate; border-spacing: 10px }

table.two { border-collapse: separate; border-spacing: 10px 50px } </style> </head>

<body> <table class="one" border="1"> <tr> <td>Peter</td> <td>Griffin</td> </tr>

<tr> <td>Lois</td> <td>Griffin</td> </tr> </table>

<table class="two" border="1"> <tr> <td>Cleveland</td> <td>Brown</td> </tr> <tr>

<td>Glenn</td> <td>Quagmire</td> </tr> </table> </body> </html>

Set the position of the table caption
This example demonstrates how to position the table caption.

<html> <head> <style type="text/css">

Caption { caption-side:bottom } </style> </head>

<body> <table border="1"> <caption>This is a caption</caption> <tr> <td>Peter</td>
<td>Griffin</td> </tr> <tr> <td>Lois</td> <td>Griffin</td> </tr> </table> </body> </html>

CSS Table Properties

The CSS table properties allow you to set the layout of a table.

Browser support: IE: Internet Explorer, M: Mac IE only, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
border-collapse	Sets whether the table borders are collapsed into a single border or detached as in standard HTML	collapse separate	5	1	7	2
border-spacing	Sets the distance that separates cell borders (only for the "separated borders" model)	*length length*	5M	1	6	2
caption-side	Sets the position of the table caption	top bottom left right	5M	1	6	2
empty-cells	Sets whether or not to show empty cells in a table (only for the "separated borders" model)	show hide	5M	1	6	2
table-layout	Sets the algorithm used to display the table cells, rows, and columns	auto fixed	5	1	6	2

CSS Dimension

The CSS dimension properties allow you to control the height and width of an element. It also allows you to increase the space between two lines.

Examples

Set the height of an image using a pixel value
This example demonstrates how to set the height of an element using a pixel value.

<html> <head> <style type="text/css">

img.normal { height: auto }

img.big { height: 160px }

img.small { height: 30px } </style> </head>

<body>

 </body> </html>

Set the height of an image using percent
This example demonstrates how to set the height of an element using a percent value.

```
<html> <head> <style type="text/css">

img.normal  { height: auto }

img.big  { height: 50% }

img.small  { height: 10% } </style> </head>

<body>  <img class="normal" src="logocss.gif" width="95" height="84" /> <br />

<img class="big" src="logocss.gif" width="95" height="84" /> <br />

<img class="small" src="logocss.gif" width="95" height="84" /> </body> </html>
```

Set the width of an element using a pixel value

This example demonstrates how to set the width of an element using a pixel value.

```
<html><head><style type="text/css">

img { width: 200px }  </style> </head>

<body>  <img src="logocss.gif" width="95" height="84" />  </body> </html>
```

Set the width of an element using percent

This example demonstrates how to set the width of an element using a percent value.

```
<html> <head> <style type="text/css">

img { width: 50% } </style> </head>

<body> <img src="logocss.gif" width="95" height="84" /> </body> </html>
```

Set the maximum height of an element

This example demonstrates how to set the maximum height of an element.

```
<html><head><style type="text/css">

p{max-height: 10px}</style></head>

<body>  <p>This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text.</p>

<img src="logocss.gif" width="95" height="84" /> </body> </html>
```

Set the maximum width of an element using a pixel value

This example demonstrates how to set the maximum width of an element using a pixel value.

```
<html> <head> <style type="text/css"> p { max-width: 300px } </style> </head>

<body>  <p>This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text.</p> </body> </html>
```

Set the maximum width of an element using percent
This example demonstrates how to set the maximum width of an element using a percent value.

```
<html> <head> <style type="text/css">

P { max-width: 50% } </style> </head>

<body>  <p>This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text.</p> </body> </html>
```

Set the minimum height of an element
This example demonstrates how to set the minimum height of an element.

```
<html> <head> <style type="text/css">

P { min-height: 100px } </style> </head>

<body>  <p>This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text.</p>

<img src="logocss.gif" width="95" height="84" />  </body> </html>
```

Set the minimum width of an element using a pixel value
This example demonstrates how to set the minimum width of an element using a pixel value.

```
<html> <head> <style type="text/css"> p { min-width: 1000px } </style> </head>

<body>  <p>This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text.</p>

<img src="logocss.gif" width="95" height="84" /> </body> </html>
```

Set the minimum width of an element using percent
This example demonstrates how to set the minimum width of an element using a percent value.

```
<html> <head> <style type="text/css"> p { min-width: 200% } </style> </head>

<body>  <p>This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text. This is some text.</p>

<img src="logocss.gif" width="95" height="84" />  </body> </html>
```

Specify the space between lines using a percent value
This example demonstrates how to specify the space between the lines in a paragraph using a percent value.

```
<html> <head> <style type="text/css"> p.small {line-height: 90%} p.big {line-height: 200%}

</style> </head>

<body> <p> This is a paragraph with a standard line-height. The default line height in most browsers is about 110% to 120%. This is a paragraph with a standard line-height. This is a paragraph with a standard line-height.</p>
```

 This is a paragraph with a smaller line-height. This is a paragraph with a smaller line-height. This is a paragraph with a smaller line-height. This is a paragraph with a smaller line-height.

 This is a paragraph with a bigger line-height. This is a paragraph with a bigger line-height. This is a paragraph with a bigger line-height. This is a paragraph with a bigger line-height.

 </body> </html>

Specify the space between lines using a pixel value

This example demonstrates how to specify the space between the lines in a paragraph using a pixel value.

<html> <head> <style type="text/css"> p.small { line-height: 10px }

p.big { line-height: 30px } </style> </head>

<body> This is a paragraph with a standard line-height. The default line height in most browsers is about 20px. This is a paragraph with a standard line-height. This is a paragraph with a standard line-height.

 This is a paragraph with a smaller line-height. This is a paragraph with a smaller line-height. This is a paragraph with a smaller line-height. This is a paragraph with a smaller line-height.

 This is a paragraph with a bigger line-height. This is a paragraph with a bigger line-height. This is a paragraph with a bigger line-height. This is a paragraph with a bigger line-height.

 </body> </html>

Specify the space between lines using a number value

This example demonstrates how to specify the space between the lines in a paragraph using a number value.

<html> <head> <style type="text/css"> p.small { line-height: 0.5 } p.big { line-height: 2 }

</style> </head> <body>

 This is a paragraph with a standard line-height. The default line height in browsers is "1".

This is a paragraph with a standard line-height. This is a paragraph with a standard line-height.

  This is a paragraph with a smaller line-height.

This is a paragraph with a smaller line-height. This is a paragraph with a smaller line-height.This is a paragraph with a smaller line-height.

 This is a paragraph with a bigger line-height. This is a paragraph with a bigger line-height. This is a paragraph with a bigger line-height. This is a paragraph with a bigger line-height.

 </body> </html>

CSS Dimension Properties

The CSS dimension properties allow you to control the height and width of an element. It also allows you to increase the space between two lines.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
height	Sets the height of an element	auto *length* *%*	4	1	6	1
line-height	Sets the distance between lines	normal *number* *length* *%*	4	1	4	1
max-height	Sets the maximum height of an element	none *length* *%*	-	1	6	2
max-width	Sets the maximum width of an element	none *length* *%*	-	1	6	2
min-height	Sets the minimum height of an element	*length* *%*	-	1	6	2
min-width	Sets the minimum width of an element	*length* *%*	-	1	6	2
width	Sets the width of an element	auto *%* *length*	4	1	4	1

CSS Classification

The CSS classification properties allow you to specify how and where to display an element.

Examples

How to display an element as an inline element.
This example demonstrates how to display an element as an inline element.

<html> <head> <style type="text/css"> p {display: inline} div {display: none} </style> </head>

<body> A display property with a value of "inline" results in

no distance between two elements. And this div section is not displayed!

</body> </html>

How to display an element as a block element
This example demonstrates how to display an element as a block element.

```
<html> <head> <style type="text/css">

Span { display: block } </style> </head>

<body> <span>A display property with a value of "block" results in</span> <span>distance between two elements.</span>  </body>  </html>
```

A simple use of the float property
Let an image float to the right in a paragraph.

```
<html> <head> <style type="text/css">

img  { float:right } </style> </head>

<body> <p>In the paragraph below, we have added an image with style <b>float:right</b>. The result is that the image will float to the right in the paragraph.</p>

<p> <img src="logocss.gif" width="95" height="84" />

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text. </p> </body>  </html>
```

An image with border and margins that floats to the right in a paragraph
Let an image float to the right in a paragraph. Add border and margins to the image.

```
<html> <head> <style type="text/css">

img  { float:right; border:1px dotted black; margin:0px 0px 15px 20px; } </style>  </head>

<body> <p>  In the paragraph below, the image will float to the right. A dotted black border is added to the image. We have also added margins to the image to push the text away from the image:
```

0 px margin on the top and right side, 15 px margin on the bottom, and 20 px margin on the left side of the image.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text. </body> </html>

An image with a caption that floats to the right
Let an image with a caption float to the right.

<html> <head> <style type="text/css">

Div { float:right; width:120px; margin:0 0 15px 20px; padding:15px; border:1px solid black;

text-align:center; } </style> </head>

<body>
 CSS is fun!

 This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

  In the paragraph above, the div element is 120 pixels wide and it contains the image.

The div element will float to the right. Margins are added to the div to push the text away from the div.

Borders and padding are added to the div to frame in the picture and the caption.   </body>
</html>

Let the first letter of a paragraph float to the left
Let the first letter of a paragraph float to the left and style the letter.

<html> <head> <style type="text/css">

span{ float:left; width:0.7em; font-size:400%; font-family:algerian,courier; line-height:80%; }

</style> </head>

<body> This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

This is some text. This is some text. This is some text.

 In the paragraph above, the first letter of the text is embedded in a span element.

The span element has a width that is 0.7 times the size of the current font.

The font-size of the span element is 400% (quite large) and the line-height is 80%.

The font of the letter in the span will be in "Algerian".   </body> </html>

Creating a horizontal menu
Use float with a list of hyperlinks to create a horizontal menu.

<html> <head> <style type="text/css">

Ul { float:left; width:100%; padding:0; margin:0; list-style-type:none; }

```
a { float:left; width:6em; text-decoration:none; color:white; background-color:purple;

padding:0.2em 0.6em; border-right:1px solid white; }

a:hover {background-color:#ff3300}

li {display:inline} </style> </head>

<body> <ul> <li><a href="#">Link one</a></li> <li><a href="#">Link two</a></li>

<li><a href="#">Link three</a></li> <li><a href="#">Link four</a></li> </ul>

<p> In the example above, we let the ul element and the a element float to the left.

The li elements will be displayed as inline elements (no line break before or after the element). This forces the list to be on one line.

The ul element has a width of 100% and each hyperlink in the list has a width of 6em (6 times the size of the current font).

We add some colors and borders to make it more fancy. </p>  </body> </html>
```

Creating a homepage without tables

Use float to create a homepage with a header, footer, left content and main content.

```
<html> <head> <style type="text/css">

div.container { width:100%; margin:0px; border:1px solid gray; line-height:150%; }

div.header,div.footer { padding:0.5em; color:white; background-color:gray; clear:left; }

h1.header { padding:0; margin:0; }

div.left { float:left; width:160px; margin:0; padding:1em; }

div.content { margin-left:190px; border-left:1px solid gray; padding:1em; } </style> </head>

<body>  <div class="container">

<div class="header"><h1 class="header">W3Schools.com</h1></div>

<div class="left"><p>"Never increase, beyond what is necessary, the number of entities required to explain anything." William of Ockham (1285-1349)</p></div>

<div class="content"> <h2>Free Web Building Tutorials</h2>

<p>At W3Schools you will find all the Web-building tutorials you need,

from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.</p>

<p>W3Schools - The Largest Web Developers Site On The Net!</p></div>

<div class="footer">Copyright 1999-2005 by Refsnes Data.</div>  </div>  </body> </html>
```

Position:relative
This example demonstrates how to position an element relative to its normal position.

<html> <head> <style type="text/css"> h2.pos_left { position:relative; left:-20px }

h2.pos_right { position:relative; left:20px } </style> </head>

<body> <h2>This is a heading in normal position</h2>

<h2 class="pos_left">This heading is moved left to its normal position</h2>

<h2 class="pos_right">This heading is moved right to its normal position</h2>

Relative positioning moves an element RELATIVE to its original position.

The style "left:-20px" subtracts 20 pixels from the element's original left position.

The style "left:20px" adds 20 pixels to the element's original left position. </body> </html>

Position:absolute
This example demonstrates how to position an element using an absolute value.

<html> <head> <style type="text/css">

h2.pos_abs { position:absolute; left:100px; top:150px } </style> </head>

<body> <h2 class="pos_abs">This is a heading with an absolute position</h2>

With absolute positioning, an element can be placed anywhere on a page. The heading below is placed 100px from the left of the page and 150px from the top of the page. </body> </html>

How to make an element invisible
This example demonstrates how to make an element invisible. Do you want the element to show or not?

<html> <head> <style type="text/css"> h1.visible {visibility:visible} h1.invisible {visibility:hidden}

</style> </head>

<body> <h1 class="visible">This is a visible heading</h1>

<h1 class="invisible">This is an invisible heading</h1> </body> </html>

How to make a table element collapse
This example demonstrates how to make a table element collapse.

<html> <head> <style type="text/css">

tr.coll { visibility:collapse } </style> </head>

<body> <table border="1"> <tr> <td>Peter</td> <td>Griffin</td> </tr> <tr class="coll">

<td>Lois</td> <td>Griffin</td> </tr> </table> </body> </html>

Change the cursor

This example demonstrates how to change the cursor.

```
<html>  <body> <p>Move the mouse over the words to see the cursor change:</p>

<span style="cursor:auto"> Auto</span><br /> <span style="cursor:crosshair">

Crosshair</span><br /> <span style="cursor:default"> Default</span><br />

<span style="cursor:pointer"> Pointer</span><br /> <span style="cursor:move">

Move</span><br /> <span style="cursor:e-resize"> e-resize</span><br />

<span style="cursor:ne-resize"> ne-resize</span><br /> <span style="cursor:nw-resize">

nw-resize</span><br /> <span style="cursor:n-resize"> n-resize</span><br />

<span style="cursor:se-resize"> se-resize</span><br /> <span style="cursor:sw-resize">

sw-resize</span><br /> <span style="cursor:s-resize"> s-resize</span><br />

<span style="cursor:w-resize"> w-resize</span><br /> <span style="cursor:text">

text</span><br /> <span style="cursor:wait"> wait</span><br /> <span style="cursor:help">

help</span> </body>  </html>
```

Clear the sides of an element

This example demonstrates how clear the sides of other floating elements.

```
<html> <head> <style type="text/css">

img { float:left;clear:both; } </style> </head>

<body> <img src="logocss.gif" width="95" height="84" />

<img src="logocss.gif" width="95" height="84" /> </body> </html>
```

CSS Classification Properties

The CSS classification properties allow you to control how to display an element, set where an image will appear in another element, position an element relative to its normal position, position an element using an absolute value, and how to control the visibility of an element.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
clear	Sets the sides of an element where other floating elements are not allowed	left right both none	4	1	4	1
cursor	Specifies the type of cursor to be displayed	*url* auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help	4	1	6	2
display	Sets how/if an element is displayed	none inline block list-item run-in compact marker table inline-table table-row-group table-header-group table-footer-group table-row table-column-group table-column table-cell table-caption	4	1	4	1
float	Sets where an image or a text will appear in another element	left right none	4	1	4	1
position	Places an element in a static, relative, absolute or fixed position	static relative absolute fixed	4	1	4	2
visibility	Sets if an element should be visible or invisible	visible hidden collapse	4	1	6	2

CSS Positioning

The CSS positioning properties allows you to position an element.

Examples

Position:relative
This example demonstrates how to position an element relative to its normal position.

<html><head> <style type="text/css"> h2.pos_left { position:relative; left:-20px }

h2.pos_right { position:relative;left:20px } </style> </head>

<body> <h2>This is a heading in normal position</h2>

<h2 class="pos_left">This heading is moved left to its normal position</h2>

<h2 class="pos_right">This heading is moved right to its normal position</h2>

Relative positioning moves an element RELATIVE to its original position.

The style "left:-20px" subtracts 20 pixels from the element's original left position.

The style "left:20px" adds 20 pixels to the element's original left position. </body> </html>

Position:absolute
This example demonstrates how to position an element using an absolute value.

<html> <head> <style type="text/css"> h2.pos_abs { position:absolute; left:100px; top:150px }

</style> </head>

<body> <h2 class="pos_abs">This is a heading with an absolute position</h2>

With absolute positioning, an element can be placed anywhere on a page. The heading below is placed 100px from the left of the page and 150px from the top of the page. </body> </html>

Position:fixed
This example demonstrates how to position an element with relative to the browser window.

<html> <head> <style type="text/css">p.one { position:fixed; left:5px; top:5px; }

p.two{ position:fixed; top:30px; right:5px; } </style> </head>

<body> Some text Some more text </body> </html>

Set the shape of an element
This example demonstrates how to set the shape of an element. The element is clipped into this shape, and displayed.

<html> <head> <style type="text/css"> img { position:absolute; clip:rect(0px 50px 200px 0px) }

</style> </head>

<body> The clip property is here cutting an image:

<img border="0" src="bookasp20.gif" width="120" height="151"> </body> </html>

How to show overflow in an element using scroll

This example demonstrates how to set the overflow property to create a scroll bar when an element's content is too big to fit in a specified area.

<html> <head> <style type="text/css">

div { background-color:#00FFFF; width:150px; height:150px; overflow: scroll } </style> </head>

<body> The overflow property decides what to do if the content inside an element exceeds the given width and height properties.

 You can use the overflow property when you want to have better control of the layout. Try to change the overflow property to: visible, hidden, auto, or inherit and see what happens. The default value is visible. </body> </html>

How to hide overflow in an element

This example demonstrates how to set the overflow property to hide the content if it is too big to fit in a specified area.

<html> <head> <style type="text/css">

div { background-color:#00FFFF; width:150px; height:150px; overflow: hidden } </style> </head>

<body> The overflow property decides what to do if the content inside an element exceeds the given width and height properties.

 You can use the overflow property when you want to have better control of the layout. Try to change the overflow property to: visible, scroll, auto, or inherit and see what happens. The default value is visible. </body> </html>

How to set the browser to automatically handle overflow

This example demonstrates how to set the browser to automatically handle overflow.

<html> <head> <style type="text/css">

div { background-color:#00FFFF;width:150px; height:150px; overflow: auto } </style> </head>

<body> The overflow property decides what to do if the content inside an element exceeds the given width and height properties.

 You can use the overflow property when you want to have better control of the layout. Try to change the overflow property to: visible, hidden, scroll, or inherit and see what happens. The default value is visible. </body> </html>

Vertical alignment of an image

This example demonstrates how to set the vertical align of an image in a text.

```
<html> <head> <style type="text/css"> img.top {vertical-align:text-top}

img.bottom {vertical-align:text-bottom} </style> </head>

<body> <p> This is an  <img class="top" border="0"  src="logocss.gif" width="95" height="84" />

image inside a paragraph. </p>

<p> This is an  <img class="bottom" border="0"  src="logocss.gif" width="95" height="84" />

image inside a paragraph. </p> </body>  </html>
```

Z-index

Z-index can be used to place an element "behind" another element.

```
<html> <head> <style type="text/css">

img.x { position:absolute; left:0px; top:0px; z-index:-1 } </style> </head>

<body> <h1>This is a Heading</h1> <img class="x" src="bulbon.gif" width="100" height="180">

<p>Default z-index is 0. Z-index -1 has lower priority.</p> </body>  </html>
```

Z-index

The elements in the example above have now changed their Z-index.

```
<html> <head> <style type="text/css">

img.x { position:absolute; left:0px; top:0px; z-index:1 } </style> </head>

<body> <h1>This is a Heading</h1> <img class="x" src="bulbon.gif" width="100" height="180">

<p>Default z-index is 0. Z-index 1 has higher priority.</p>  </body>  </html>
```

Set the top edge of an image using a pixel value

This example demonstrates how to set the top edge of an element using a pixel value.

```
<html> <head> <style type="text/css">

img { position:absolute; top:0px } </style> </head>

<body>  <h1>This is a Heading</h1> <img src="logocss.gif" width="95" height="84" />

<p>Some text.</p>  </body> </html>
```

Set the left edge of an image using a percent value

This example demonstrates how to set the left edge of an element using a percent value.

```
<html> <head> <style type="text/css">img { position:absolute; left:5% } </style> </head>

<body>  <h1>This is a Heading</h1> <img src="logocss.gif" width="95" height="84" />

<p>Some text.</p>  </body> </html>
```

CSS Positioning Properties

The CSS positioning properties allow you to specify the left, right, top, and bottom position of an element. It also allows you to set the shape of an element, place an element behind another, and to specify what should happen when an element's content is too big to fit in a specified area.

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Property	Description	Values	IE	F	N	W3C
bottom	Sets how far the bottom edge of an element is above/below the bottom edge of the parent element	auto *%* *length*	5	1	6	2
clip	Sets the shape of an element. The element is clipped into this shape, and displayed	*shape* auto	4	1	6	2
left	Sets how far the left edge of an element is to the right/left of the left edge of the parent element	auto *%* *length*	4	1	4	2
overflow	Sets what happens if the content of an element overflow its area	visible hidden scroll auto	4	1	6	2
position	Places an element in a static, relative, absolute or fixed position	static relative absolute fixed	4	1	4	2
right	Sets how far the right edge of an element is to the left/right of the right edge of the parent element	auto *%* *length*	5	1	6	2
top	Sets how far the top edge of an element is above/below the top edge of the parent element	auto *%* *length*	4	1	4	2
vertical-align	Sets the vertical alignment of an element	baseline sub super top text-top middle bottom text-bottom *length* *%*	4	1	4	1
z-index	Sets the stack order of an element	auto *number*	4	1	6	2

CSS Pseudo-classes

CSS pseudo-classes are used to add special effects to some selectors.

Examples

Hyperlink
This example demonstrates how to add different colors to a hyperlink in a document.

<html> <head> <style type="text/css"> a:link {color: #FF0000} a:visited {color: #00FF00}

a:hover {color: #FF00FF} a:active {color: #0000FF} </style> </head>

<body> This is a link

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!!

Note: a:active MUST come after a:hover in the CSS definition in order to be effective!! </body> </html>

Hyperlink 2
This example demonstrates how to add other styles to hyperlinks.

<html> <head> <style type="text/css"> a.one:link {color: #ff0000} a.one:visited {color: #0000ff}

a.one:hover {color: #ffcc00} a.two:link {color: #ff0000} a.two:visited {color: #0000ff}

a.two:hover {font-size: 150%} a.three:link {color: #ff0000} a.three:visited {color: #0000ff}

a.three:hover {background: #66ff66} a.four:link {color: #ff0000} a.four:visited {color: #0000ff}

a.four:hover {font-family: monospace} a.five:link {color: #ff0000; text-decoration: none}

a.five:visited {color: #0000ff; text-decoration: none} a.five:hover {text-decoration: underline}

</style> </head>

<body> Mouse over the links to see them change layout.

This link changes color

This link changes font-size

This link changes background-color

This link changes font-family

This link changes text-decoration </body> </html>

Hyperlink: use of :focus (does not work in IE)
This example demonstrates how to use the :focus pseudo-class on a hyperlink.

<html> <head> <style type="text/css"> a:link {color: #FF0000} a:focus {color: #00FF00} </style>

</head>

<body> This is a link </body> </html>

**:first-child - change first child **
This example sets any element that is the first child of any element to bold.
(for IE <!DOCTYPE> must be declared)

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html> <head> <style type="text/css"> p:first-child { font-weight:bold } </style> </head>

<body> I am a strong man. I am a strong man.

I am a strong man. I am a strong man. </body> </html>

:first-child - change first child in all elements
This example sets the first element in all elements to bold.
(for IE <!DOCTYPE> must be declared)

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html> <head> <style type="text/css"> p > em:first-child { font-weight:bold } </style> </head>

<body> I am a strong man. I am a strong man.

I am a strong man. I am a strong man. </body> </html>

**:first-child - change all elements in first child **
This example sets any elements in first child elements to bold.
(for IE <!DOCTYPE> must be declared)

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html> <head> <style type="text/css"> p:first-child em { font-weight:bold } </style> </head>

<body> I am a strong man. I am a strong man.

I am a strong man. I am a strong man. </body> </html>

:lang (does not work in IE)
This example demonstrates the use of the :lang pseudo-class.

<html><head> <style type="text/css"> q:lang(no) { quotes: "~" "~" } </style> </head>

<body>

The :lang pseudo-class allows you to define special rules for different languages. In the example below, the :lang class defines the type of quotation marks for q elements with a lang attribute with a value of "no":

Some text <q lang="no">A quote in a paragraph</q> Some text. </body> </html>

Syntax

The syntax of pseudo-classes:

```
selector:pseudo-class {property: value}
```

CSS classes can also be used with pseudo-classes:

```
selector.class:pseudo-class {property: value}
```

Anchor Pseudo-classes

A link that is active, visited, unvisited, or when you mouse over a link can all be displayed in different ways in a CSS-supporting browser:

```
a:link {color: #FF0000}     /* unvisited link */
a:visited {color: #00FF00}  /* visited link */
a:hover {color: #FF00FF}    /* mouse over link */
a:active {color: #0000FF}   /* selected link */
```

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!!

Note: a:active MUST come after a:hover in the CSS definition in order to be effective!!

Note: Pseudo-class names are not case-sensitive.

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

```
a.red:visited {color: #FF0000}
<a class="red" href="css_syntax.asp">CSS Syntax</a>
```

If the link in the example above has been visited, it will be displayed in red.

CSS2 - The :first-child Pseudo-class

The :first-child pseudo-class matches a specified element that is the first child of another element.

Note: For :first-child to work in IE. A <!DOCTYPE> must be declared.

Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

```
<html>
<head>
<style type="text/css">
p:first-child
{
font-weight:bold
}
</style>
</head>
<body>
<p>I am a <em>strong</em> man. I am a <em>strong</em> man.</p>
<p>I am a <em>strong</em> man. I am a <em>strong</em> man.</p>
</body>
</html>
```

Match the first element in all <p> elements

In the following example, the selector matches any element that is the first child of a <p> element:

```
<html>
<head>
<style type="text/css">
p > em:first-child
{
font-weight:bold
}
</style>
</head>
<body>
<p>I am a <em>strong</em> man. I am a <em>strong</em> man.</p>
<p>I am a <em>strong</em> man. I am a <em>strong</em> man.</p>
</body>
</html>
```

Match all elements in all first child <p> elements

In the following example, the selector matches any elements in <p> elements that are the first child of another element:

```
<html>
<head>
<style type="text/css">
p:first-child em
{
font-weight:bold
}
</style>
</head>
<body>
<p>I am a <em>strong</em> man. I am a <em>strong</em> man.</p>
<p>I am a <em>strong</em> man. I am a <em>strong</em> man.</p>
</body>
</html>
```

CSS2 - The :lang Pseudo-class

The :lang pseudo-class allows you to define special rules for different languages. In the example below, the :lang class defines the type of quotation marks for q elements with a lang attribute with a value of "no":

```
<html>
<head>
<style type="text/css">
q:lang(no)
{
quotes: "~" "~"
}
</style>
</head>
<body>
<p>Some text <q lang="no">A quote in a paragraph</q>
Some text.</p>
</body>
</html>
```

Pseudo-classes

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Pseudo-class	Purpose	IE	F	N	W3C
:active	Adds special style to an activated element	4	1	8	1
:focus	Adds special style to an element while the element has focus	-	1.5	8	2
:hover	Adds special style to an element when you mouse over it	4	1	7	1
:link	Adds special style to an unvisited link	3	1	4	1
:visited	Adds special style to a visited link	3	1	4	1
:first-child	Adds special style to an element that is the first child of some other element	7	1	7	2
:lang	Allows the author to specify a language to use in a specified element	-	1	8	2

CSS Pseudo-elements

CSS pseudo-elements are used to add special effects to some selectors.

Examples

Make the first letter special
This example demonstrates how to add a special effect to the first letter of a text.

<html> <head> <style type="text/css"> p:first-letter { color: #ff0000; font-size:xx-large }

</style> </head>

<body> You can use the :first-letter pseudo-element to add a special effect to the first letter of a text! </body> </html>

Make the first line special
This example demonstrates how to add a special effect to the first line of a text.

<html> <head> <style type="text/css"> p:first-line { color: #ff0000; font-variant: small-caps }

</style> </head>

<body> You can use the :first-line pseudo-element to add a special effect to the first line of a text! </body> </html>

Make the first letter and first line special
This example demonstrates how to add a special effect to the first letter and the first line of a text.

<html> <head> <style type="text/css">

p:first-letter { color: #ff0000; font-size:xx-large }

p:first-line { color: #0000ff; font-variant: small-caps } </style> </head>

<body> You can combine the :first-letter and :first-line pseudo-elements to add a special effect to the first letter and the first line of a text! </body> </html>

Use :before to insert some content before the content of an element (Does not work in IE)
This example demonstrates how to use the :before pseudo-element to insert an image before an element.

<html> <head> <style type="text/css"> h1:before { content: url(smiley.gif) } </style> </head>

<body> <h1>This is a header</h1> The :before pseudo-element can be used to insert some content before the content of an element. </body> </html>

Use :after to insert some content after the content of an element (Does not work in IE)
This example demonstrates how to use the :after pseudo-element to insert an image after an element.

<html> <head> <style type="text/css"> h1:after { content: url(smiley.gif) } </style> </head>

<body> <h1>This is a header</h1> The :after pseudo-element can be used to insert some content after the content of an element. </body> </html>

Syntax

The syntax of pseudo-elements:

```
selector:pseudo-element {property: value}
```

CSS classes can also be used with pseudo-elements:

```
selector.class:pseudo-element {property: value}
```

The :first-line Pseudo-element

The "first-line" pseudo-element is used to add special styles to the first line of the text in a selector:

```
p:first-line {color:#0000ff;font-variant:small-caps}
<p>Some text that ends up on two or more lines</p>
```

The output could be something like this:

```
Some text that ends
up on two or more lines
```

In the example above the browser displays the first line formatted according to the "first-line" pseudo element. **Where** the browser breaks the line depends on the size of the browser window.

Note: The "first-line" pseudo-element can only be used with block-level elements.

Note: The following properties apply to the "first-line" pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

The :first-letter Pseudo-element

The "first-letter" pseudo-element is used to add special style to the first letter of the text in a selector:

```
p:first-letter {color:#ff0000;font-size:xx-large}
<p>The first words of an article...</p>
```

The output could be something like this:

```
The first words of an article...
```

Note: The "first-letter" pseudo-element can only be used with block-level elements.

Note: The following properties apply to the "first-letter" pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

```
p.article:first-letter {color:#ff0000}
<p class="article">A paragraph in an article</p>
```

The example above will make the first letter of all paragraphs with class="article" red.

Multiple Pseudo-elements

Several pseudo-elements can be combined:

```
p:first-letter {color:#ff0000;font-size:xx-large}
p:first-line {color:#0000ff}
<p>The first words of an article...</p>
```

The output could be something like this:

```
The first
words of an
article...
```

In the example above the first letter of the paragraph will be red with a font size of 24pt. The rest of the first line would be blue while the rest of the paragraph would be the default color.

CSS2 - The :before Pseudo-element

The ":before" pseudo-element can be used to insert some content before the content of an element.

The style below will play a sound before each occurrence of an <h1> element:

```
h1:before
{
content: url(beep.wav)
}
```

CSS2 - The :after Pseudo-element

The ":after" pseudo-element can be used to insert some content after the content of an element.

The style below will play a sound after each occurrence of an <h1> element:

```
h1:after
{
content: url(beep.wav)
}
```

Pseudo-elements

Browser support: IE: Internet Explorer, F: Firefox, N: Netscape.

W3C: The number in the "W3C" column indicates in which CSS recommendation the property is defined (CSS1 or CSS2).

Pseudo-element	Purpose	IE	F	N	W3C
:first-letter	Adds special style to the first letter of a text	5	1	8	1
:first-line	Adds special style to the first line of a text	5	1	8	1
:before	Inserts some content before the content of an element		1.5	8	2
:after	Inserts some content after the content of an element		1.5	8	2

CSS Image Gallery

CSS can be used to create an image gallery.

Image Gallery

The following image gallery is created with CSS:

Image gallery

The source code looks like this:

```
<html>
<head>
<style type="text/css">
div.img
  {
  margin: 2px;
  border: 1px solid #0000ff;
  height: auto;
  width: auto;
  float: left;
  text-align: center;
  }
div.img img
  {
  display: inline;
  margin: 3px;
  border: 1px solid #ffffff;
  }
div.img a:hover img
  {
  border: 1px solid #0000ff;
  }
div.desc
  {
  text-align: center;
  font-weight: normal;
  width: 120px;
  margin: 2px;
  }
</style>
</head>
```

```
<body>
<div class="img">
  <a target="_blank" href="klematis_big.htm">
  <img src="klematis_small.jpg" alt="Klematis" width="110"
 height="90" />
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="klematis2_big.htm">
  <img src="klematis2_small.jpg" alt="Klematis" width="110"
 height="90" />
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="klematis3_big.htm">
  <img src="klematis3_small.jpg" alt="Klematis" width="110"
 height="90" />
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="klematis4_big.htm">
  <img src="klematis4_small.jpg" alt="Klematis" width="110"
 height="90" />
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
</body>
</html>
```

CSS Image Opacity / Transparency

Creating transparent images with CSS is easy.

Example 1 - Creating a Transparent Image

First we will show you how to create a transparent image with CSS.

Regular image:

The same image with transparency:

Look at the following source code:

```
<img src="klematis.jpg" width="150" height="113" alt="klematis"
style="opacity:0.4;filter:alpha(opacity=40)" />
```

Firefox uses the property **opacity:x** for transparency, while IE uses **filter:alpha(opacity=x)**.

Tip: The CSS3 syntax for transparency is **opacity:x**.

In Firefox (opacity:x) x can be a value from 0.0 - 1.0. A lower value makes the element more transparent.

In IE (filter:alpha(opacity=x)) x can be a value from 0 - 100. A lower value makes the element more transparent.

Example 2 - Image Transparency - Mouseover Effect

Mouse over the images:

The source code looks like this:

```
<img src="klematis.jpg" style="opacity:0.4;filter:alpha(opacity=40)"
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />
<img src="klematis2.jpg" style="opacity:0.4;filter:alpha(opacity=40)"
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />
```

We see that the first line of the source code is similar to the source code in Example 1. In addition, we have added an onmouseover attribute and an onmouseout attribute. The onmouseover attribute defines what will happen when the mouse pointer moves over the image. In this case we want the image to NOT be transparent when we move the mouse pointer over it.

The syntax for this in Firefox is: **this.style.opacity=1** and the syntax in IE is: **this.filters.alpha.opacity=100**.

When the mouse pointer moves away from the image, we want the image to be transparent again. This is done in the onmouseout attribute.

Example 3 - Text in Transparent Box

This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box.

The source code looks like this:

```
<html>
<head>
<style type="text/css">
div.background
  {
  width: 500px;
  height: 250px;
  background: url(klematis.jpg) repeat;
  border: 2px solid black;
  }
div.transbox
  {
  width: 400px;
  height: 180px;
  margin: 30px 50px;
  background-color: #ffffff;
  border: 1px solid black;
  /* for IE */
  filter:alpha(opacity=60);
  /* CSS3 standard */
  opacity:0.6;
  }
div.transbox p
  {
  margin: 30px 40px;
  font-weight: bold;
  color: #000000;
  }
</style>
</head>
<body>
<div class="background">
<div class="transbox">
<p>This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
</p>
</div>
</div>
```

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

First, we create a div element (class="background") with a fixed height and width, a background image, and a border. Then we create a smaller div (class="transbox") inside the first div element. This div also have a fixed width, a background image, and a border. In addition we make this div transparent.

Inside the transparent div, we add some text inside a p element.

CSS2 Media Types

Media Types allow you to specify how documents will be presented in different media. The document can be displayed differently on the screen, on the paper, with an aural browser, etc.

Media Types

Some CSS properties are only designed for a certain media. For example the "voice-family" property is designed for aural user agents. Some other properties can be used for different media types. For example, the "font-size" property can be used for both screen and print media, but perhaps with different values. A document usually needs a larger font-size on a screen than on paper, and sans-serif fonts are easier to read on the screen, while serif fonts are easier to read on paper.

The @media Rule

The @media rule allows different style rules for different media in the same style sheet.

The style in the example below tells the browser to display a 14 pixels Verdana font on the screen. But if the page is printed, it will be in a 10 pixels Times font. Notice that the font-weight is set to bold, both on screen and on paper:

```
<html>
<head>
<style>
@media screen
{
p.test {font-family:verdana,sans-serif; font-size:14px}
}

@media print
{
p.test {font-family:times,serif; font-size:10px}
}
@media screen,print
{
p.test {font-weight:bold}
}
```

```
</style>
</head>
<body>
....
</body>
</html>
```

See it yourself ! If you are using Mozilla/Firefox or IE 5+ and print this page, you will see that the paragraph under "Media Types" will be displayed in another font, and have a smaller font size than the rest of the text.

Different Media Types

Note: The media type names are not case-sensitive.

Media Type	Description
all	Used for all media type devices
aural	Used for speech and sound synthesizers
braille	Used for braille tactile feedback devices
embossed	Used for paged braille printers
handheld	Used for small or handheld devices
print	Used for printers
projection	Used for projected presentations, like slides
screen	Used for computer screens
tty	Used for media using a fixed-pitch character grid, like teletypes and terminals
tv	Used for television-type devices

CSS Don't

Here are some technologies you should try to avoid when using CSS.

Internet Explorer Behaviors

What is it? Internet Explorer 5 introduced behaviors. Behaviors are a way to add behaviors to HTML elements with the use of CSS styles.

Why avoid it? The behavior attribute is only supported by Internet Explorer.

What to use instead? Use JavaScript and HTML DOM instead.

Example 1 - Mouseover Highlight

The following HTML file has a <style> element that defines a behavior for the <h1> element:

```
<html>
<head>
<style type="text/css">
h1 { behavior: url(behave.htc) }
</style>
</head>

<body>
<h1>Mouse over me!!!</h1>
</body>
</html>
```

The XML document "behave.htc" is shown below:

```
<attach for="element" event="onmouseover" handler="hig_lite" />
<attach for="element" event="onmouseout" handler="low_lite" />

<script type="text/javascript">
function hig_lite()
{
element.style.color='red';
}
function low_lite()
{
element.style.color='blue';
}
</script>
```

The behavior file contains a JavaScript and event handlers for the elements.

If you use Internet Explorer, try it yourself (mouse over the text in the example).

Example 2 - Typewriter Simulation

The following HTML file has a <style> element that defines a behavior for elements with an id of "typing":

```
<html>
<head>
<style type="text/css">
#typing
{
behavior:url(behave_typing.htc);
font-family:"courier new";
}
</style>
</head>

<body>
<span id="typing" speed="100">IE5 introduced DHTML behaviors.
Behaviors are a way to add DHTML functionality to HTML elements
with the ease of CSS.<br /><br />How do behaviors work?<br />
By using XML we can link behaviors to any element in a web page
and manipulate that element.</p>
```

```
</span>
</body>
</html>
```

The XML document "typing.htc" is shown below:

```
<attach for="window" event="onload" handler="beginTyping" />
<method name="type" />
<script type="text/javascript">
var i,text1,text2,textLength,t;
function beginTyping()
{
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
t=window.setInterval(element.id+".type()",speed);
}
function type()
{
text2=text2+text1.substring(i,i+1);
element.innerText=text2;
i=i+1;
if (i==textLength)
  {
  clearInterval(t);
  }
}
</script>
```

You Have Learned CSS, Now What?

CSS Summary

This tutorial has taught you how to create style sheets to control the style and layout of multiple web sites at once.

You have learned how to use CSS to add backgrounds, format text, add and format borders, and specify padding and margins of elements.

You have also learned how to position an element, control the visibility and size of an element, set the shape of an element, place an element behind another, and to add special effects to some selectors, like links.

JavaScript Tutorial

JavaScript Tutorial

JavaScript is *THE* scripting language of the Web.

JavaScript is used in millions of Web pages to add functionality, validate forms, detect browsers, and much more.

JavaScript is easy to learn! You will enjoy it!

Start learning JavaScript now!

Introduction to JavaScript

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, and Opera.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML / XHTML

If you want to study these subjects first, find the tutorials on our Home page.

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Are Java and JavaScript the Same?

NO!

Java and JavaScript are two completely different languages in both concept and design!

Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript Do?

- **JavaScript gives HTML designers a programming tool -** HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- **JavaScript can put dynamic text into an HTML page -** A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

---

The Real Name is ECMAScript

JavaScript's official name is "ECMAScript". The standard is developed and maintained by the ECMA organisation.

ECMA-262 is the official JavaScript standard. The standard is based on JavaScript (Netscape) and JScript (Microsoft).

The language was invented by Brendan Eich at Netscape (with Navigator 2.0), and has appeared in all Netscape and Microsoft browsers since 1996.

The development of ECMA-262 started in 1996, and the first edition of was adopted by the ECMA General Assembly in June 1997.

The standard was approved as an international ISO (ISO/IEC 16262) standard in 1998.

The development of the standard is still in progress.

JavaScript How To ...

The HTML <script> tag is used to insert a JavaScript into an HTML page.

Examples

Write text with Javascript
The example demonstrates how to use JavaSript to write text on a web page.

Write HTML with Javascript

The example demonstrates how to use JavaScript to write HTML tags on a web page.

How to Put a JavaScript Into an HTML Page

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!");
</script>
</body>
</html>
```

The code above will produce this output on an HTML page:

```
Hello World!
```

Example Explained

To insert a JavaScript into an HTML page, we use the <script> tag. Inside the <script> tag we use the type attribute to define the scripting language.

So, the <script type="text/javascript"> and </script> tells where the JavaScript starts and ends:

```
<html>
<body>
<script type="text/javascript">
...
</script>
</body>
</html>
```

The word **document.write** is a standard JavaScript command for writing output to a page.

By entering the document.write command between the <script> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page:

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!");
</script>
</body>
</html>
```

Note: If we had not entered the <script> tag, the browser would have treated the document.write("Hello World!") command as pure text, and just write the entire line on the page.

HTML Comments to Handle Simple Browsers

Browsers that do not support JavaScript will display JavaScript as page content.

To prevent them from doing this, and as a part of the JavaScript standard, the HTML comment tag can be used to "hide" the JavaScript. Just add an HTML comment tag <!-- before the first JavaScript statement, and a --> (end of comment) after the last JavaScript statement.

```
<html>
<body>
<script type="text/javascript">
<!--
document.write("Hello World!");
//-->
</script>
</body>
</html>
```

The two forward slashes at the end of comment line (//) is the JavaScript comment symbol. This prevents JavaScript from executing the --> tag.

JavaScript Where To ...

JavaScripts in the body section will be executed WHILE the page loads.

JavaScripts in the head section will be executed when CALLED.

Examples

Head section
Scripts that contain functions go in the head section of the document. Then we can be sure that the script is loaded before the function is called.

Body section
Execute a script that is placed in the body section.

External script
How to access an external script.

Where to Put the JavaScript

JavaScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

Scripts in the head section: Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
```

Scripts in the body section: Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>
```

Scripts in both the body and the head section: You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>
```

Using an External JavaScript: Sometimes you might want to run the same JavaScript on several pages, without having to write the same script on every page.

To simplify this, you can write a JavaScript in an external file. Save the external JavaScript file with a .js file extension.

Note: The external script cannot contain the <script> tag!

To use the external script, point to the .js file in the "src" attribute of the <script> tag:

```
<html>
<head>
<script src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

Note: Remember to place the script exactly where you normally would write the script!

JavaScript Statements

JavaScript is a sequence of statements to be executed by the browser.

JavaScript is Case Sensitive

Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

JavaScript Statements

A JavaScript statement is a command to the browser. The purpose of the command is to tell the browser what to do.

This JavaScript statement tells the browser to write "Hello Dolly" to the web page:

```
document.write("Hello Dolly");
```

It is normal to add a semicolon at the end of each executable statement. Most people think this is a good programming practice, and most often you will see this in JavaScript examples on the web.

The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Because of this you will often see examples without the semicolon at the end.

Note: Using semicolons makes it possible to write multiple statements on one line.

JavaScript Code

JavaScript code (or just JavaScript) is a sequence of JavaScript statements.

Each statement is executed by the browser in the sequence they are written.

This example will write a header and two paragraphs to a web page:

```
<script type="text/javascript">
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

JavaScript Blocks

JavaScript statements can be grouped together in blocks.

Blocks start with a left curly bracket {, and ends with a right curly bracket }.

The purpose of a block is to make the sequence of statements execute together.

This example will write a header and two paragraphs to a web page:

```
<script type="text/javascript">
{
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
}
</script>
```

The example above is not very useful. It just demonstrates the use of a block. Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

You will learn more about functions and conditions in later chapters.

JavaScript Comments

JavaScript comments can be used to make the code more readable.

JavaScript Comments

Comments can be added to explain the JavaScript, or to make it more readable.

Single line comments start with //.

This example uses single line comments to explain the code:

```
<script type="text/javascript">
// This will write a header:
document.write("<h1>This is a header</h1>");
// This will write two paragraphs:
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

JavaScript Multi-Line Comments

Multi line comments start with /* and end with */.

This example uses a multi line comment to explain the code:

```
<script type="text/javascript">
/*
The code below will write
one header and two paragraphs
*/
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

Using Comments to Prevent Execution

In this example the comment is used to prevent the execution of a single code line:

```
<script type="text/javascript">
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
//document.write("<p>This is another paragraph</p>");
</script>
```

In this example the comments is used to prevent the execution of multiple code lines:

```
<script type="text/javascript">
/*
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
*/
</script>
```

Using Comments at the End of a Line

In this example the comment is placed at the end of a line:

```
<script type="text/javascript">
document.write("Hello"); // This will write "Hello"
document.write("Dolly"); // This will write "Dolly"
</script>
```

JavaScript Variables

Variables are "containers" for storing information:

x=5; carname="Volvo";

Do You Remember Algebra From School?

Hopefully you remember algebra like this from school: x=5, y=6, z=x+y.

Do you remember that a letter (like x) could be used to hold a value (like 5), and that you could use the information above to calculate the value of z to be 11?

Sure you do!

These letters are called **variables**, and variables can be used to hold values (x=5) or expressions (z=x+y).

JavaScript Variables

As with algebra, JavaScript variables are used to hold values or expressions.

A variable can have a short name, like **x**, or a more descriptive name like **carname**.

Rules for JavaScript variable names:

- Variable names are case sensitive (**y** and **Y** are two different variables)
- Variable names must **begin with a letter** or the underscore character

NOTE: Because JavaScript is case-sensitive, variable names are case-sensitive.

Example

A variable's value can change during the execution of a script. You can refer to a variable by its name to display or change its value.

<html><body> <script type="text/javascript"> var firstname; firstname="Hege";

document.write(firstname);document.write("
"); firstname="Tove"; document.write(firstname);

</script>

The script above declares a variable,assigns a value to it, displays the value, change the value,

and displays the value again. </body> </html>

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var statement**:

```
var x;
var carname;
```

After the declaration shown above, the variables have no values, but you can assign values to the variables while you declare them:

```
var x=5;
var carname="Volvo";
```

Note: When you assign a text value to a variable, you use quotes around the value.

Assigning Values to JavaScript Variables

You assign values to JavaScript variables with **assignment statements**:

```
x=5;
carname="Volvo";
```

The variable name is on the left side of the = sign, and the value you want to assign to the variable is on the right.

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that have not yet been declared, the variables will automatically be declared.

These statements:

```
x=5;
carname="Volvo";
```

have the same effect as:

```
var x=5;
var carname="Volvo";
```

Redeclaring JavaScript Variables

If you redeclare a JavaScript variable, it will not lose its original value.

```
var x=5;
var x;
```

After the execution of the statements above, the variable x will still have the value of 5. The value of x is not reset (or cleared) when you redeclare it.

JavaScript Arithmetic

As with algebra, you can do arithmetic with JavaScript variables:

```
y=x-5;
z=y+5;
```

You will learn more about the operators that can be used between JavaScript variables in the next chapter of this tutorial.

JavaScript Operators

The operator = is used to assign values.

The operator + is used to add values.

The assignment operator **=** is used to assign values to JavaScript variables.

The arithmetic operator + is used to add values together.

```
y=5;
z=2;
x=y+z;
```

The value of x, after the execution of the statements above is 7.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1

++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

Look at these examples:

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

The rule is:

If you add a number and a string, the result will be a string.

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false

<=	is less than or equal to	x<=8 is true

How Can it be Used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age<18) document.write("Too young");
```

You will learn more about the use of conditional statements in the next chapter of this tutorial.

Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

```
variablename=(condition)?value1:value2
```

Example

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President " else it will be assigned "Dear".

JavaScript If...Else Statements

Conditional statements in JavaScript are used to perform different actions based on different conditions.

Examples

If statement
How to write an if statement.

<html> <body> <script type="text/javascript"> var d = new Date(); var time = d.getHours();

if (time < 10) { document.write("Good morning"); } </script>

 This example demonstrates the If statement.

 If the time on your browser is less than 10, you will get a "Good morning" greeting.

</body> </html>

If...else statement
How to write an if...else statement.

<html> <body> <script type="text/javascript"> var d = new Date(); var time = d.getHours();

if (time < 10) {document.write("Good morning"); }

else { document.write("Good day"); } </script>

This example demonstrates the If...Else statement.

 If the time on your browser is less than 10, you will get a "Good morning" greeting.

Otherwise you will get a "Good day" greeting. </body> </html>

If..else if...else statement
How to write an if..else if...else statement.

<html> <body> <script type="text/javascript"> var d = new Date(); var time = d.getHours();

if (time<10){ document.write("Good morning"); }

else if (time>=10 && time<16){ document.write("Good day"); }

else{ document.write("Hello World!"); } </script>

 This example demonstrates the if..else if...else statement. </body> </html>

Random link

This example demonstrates a link, when you click on the link it will take you to W3Schools.com OR to RefsnesData.no. There is a 50% chance for each of them.

```
<html> <body> <script type="text/javascript"> var r=Math.random();

if (r>0.5){ document.write("<a href='http://www.w3schools.com'>Learn Web Development!</a>"); }

else{ document.write("<a href='http://www.refsnesdata.no'>Visit Refsnes Data!</a>"); } </script>

</body> </html>
```

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement if you want to execute some code only if a specified condition is true
- **if...else statement** - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement if you want to select one of many blocks of code to be executed
- **switch statement** - use this statement if you want to select one of many blocks of code to be executed

---

If Statement

You should use the if statement if you want to execute some code only if a specified condition is true.

Syntax

```
if (condition)
{
code to be executed if condition is true
}
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Example 1

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10
var d=new Date();
var time=d.getHours();

if (time<10)
{
document.write("<b>Good morning</b>");
```

```
}
</script>
```

Example 2

```
<script type="text/javascript">
//Write "Lunch-time!" if the time is 11
var d=new Date();
var time=d.getHours();

if (time==11)
{
document.write("<b>Lunch-time!</b>");
}
</script>
```

Note: When **comparing** variables you must always use two equals signs next to each other (==)!

Notice that there is no ..else.. in this syntax. You just tell the code to execute some code **only if the specified condition is true**.

If...else Statement

If you want to execute some code if a condition is true and another code if the condition is not true, use the if....else statement.

Syntax

```
if (condition)
{
code to be executed if condition is true
}
else
{
code to be executed if condition is not true
}
```

Example

```
<script type="text/javascript">
//If the time is less than 10,
//you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting.
var d = new Date();
var time = d.getHours();

if (time < 10)
{
document.write("Good morning!");
}
else
{
document.write("Good day!");
}
</script>
```

If...else if...else Statement

You should use the if....else if...else statement if you want to select one of many sets of lines to execute.

Syntax

```
if (condition1)
{
code to be executed if condition1 is true
}
else if (condition2)
{
code to be executed if condition2 is true
}
else
{
code to be executed if condition1 and
condition2 are not true
}
```

Example

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
{
document.write("<b>Good morning</b>");
}
else if (time>10 && time<16)
{
document.write("<b>Good day</b>");
}
else
{
document.write("<b>Hello World!</b>");
}
</script>
```

JavaScript Switch Statement

Conditional statements in JavaScript are used to perform different actions based on different conditions.

Examples

Switch statement
How to write a switch statement.

The JavaScript Switch Statement

You should use the switch statement if you want to select one of many blocks of code to be executed.

Syntax

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is
  different from case 1 and 2
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

Example

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this weekend!");
}
</script>
```

JavaScript Popup Boxes

In JavaScript we can create three kinds of popup boxes: Alert box, Confirm box, and Prompt box.

Examples

Alert box

```
<html> <head><script type="text/javascript">

function disp_alert(){ alert("Hello again! This is how we" + '\n' + "add line breaks to an alert box!"); }

</script></head>

<body> <input type="button" onclick="disp_alert()" value="Display alert box" />  </body> </html>
```

Alert box with line breaks

```
<html><head> <script type="text/javascript">

function disp_confirm(){ var r=confirm("Press a button");

if (r==true)  {   document.write("You pressed OK!");   }

else  {   document.write("You pressed Cancel!");   } }  </script></head>

<body> <input type="button" onclick="disp_confirm()" value="Display a confirm box" />  </body>

</html>
```

Confirm box

```
<html> <head><script type="text/javascript">

function disp_prompt(){ var name=prompt("Please enter your name","Harry Potter");

if (name!=null && name!="")  {   document.write("Hello " + name + "! How are you today?");   } }

</script></head>

<body> <input type="button" onclick="disp_prompt()" value="Display a prompt box" />  </body>

</html>
```

Prompt box

```
<html><head> <script type="text/javascript"> function disp_alert()  { alert("I am an alert box!!"); }

</script></head>
```

```
<body> <input type="button" onclick="disp_alert()" value="Display alert box" />  </body> </html>
```

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax:

```
alert("sometext");
```

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax:

```
confirm("sometext");
```

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax:

```
prompt("sometext","defaultvalue");
```

JavaScript Functions

A function is a reusable code-block that will be executed by an event, or when the function is called.

Examples

Function
How to call a function.

```
<html><head>  <script type="text/javascript">

function myfunction(){ alert("HELLO"); } </script>  </head>

<body> <form> <input type="button"  onclick="myfunction()"  value="Call function"> </form>

<p>By pressing the button, a function will be called. The function will alert a message.</p>

</body></html>
```

Function with arguments
How to pass a variable to a function, and use the variable in the function.

```
<html> <head>  <script type="text/javascript">

function myfunction(txt){ alert(txt); } </script> </head>

<body> <form> <input type="button"  onclick="myfunction('Hello')" value="Call function"> </form>

<p>By pressing the button, a function with an argument will be called. The function will alert

this argument.</p> </body> </html>
```

Function with arguments 2
How to pass variables to a function, and use these variables in the function.

```
<html> <head>  <script type="text/javascript">

function myfunction(txt) {  alert(txt); }  </script>  </head>

<body>  <form>  <input type="button" onclick="myfunction('Good Morning!')"  value="In the
Morning">  <input type="button"  onclick="myfunction('Good Evening!')"  value="In the Evening">
</form>

<p> When you click on one of the buttons, a function will be called. The function will alert

the argument that is passed to it.</p>  </body>  </html>
```

Function that returns a value
How to let the function return a value.

```
<html><head>  <script type="text/javascript">

function myFunction(){ return ("Hello, have a nice day!"); } </script>  </head>

<body> <script type="text/javascript"> document.write(myFunction()) </script>

<p>The script in the body section calls a function.</p>
```

The function returns a text. </body> </html>

A function with arguments, that returns a value
How to let the function find the product of two arguments and return the result.

<html><head> <script type="text/javascript">

function product(a,b){ return a*b; } </script> </head>

<body><script type="text/javascript"> document.write(product(4,3)); </script>

The script in the body section calls a function with two parameters (4 and 3).

The function will return the product of these two parameters. </body> </html>

JavaScript Functions

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to that function.

You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that the function is read/loaded by the browser before it is called, it could be wise to put it in the <head> section.

Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!"
onclick="displaymessage()" >
</form>
</body>
</html>
```

If the line: alert("Hello world!!") in the example above had not been put within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before the user hits the button. We have added an onClick event to the button that will execute the function displaymessage() when the button is clicked.

You will learn more about JavaScript events in the JS Events chapter.

How to Define a Function

The syntax for creating a function is:

```
function functionname(var1,var2,...,varX)
{
some code
}
```

var1, var2, etc are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name:

```
function functionname()
{
some code
}
```

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

The return Statement

The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

Example

The function below should return the product of two numbers (a and b):

```
function prod(a,b)
{
x=a*b;
return x;
}
```

When you call the function above, you must pass along two parameters:

```
product=prod(2,3);
```

The returned value from the prod() function is 6, and it will be stored in the variable called product.

The Lifetime of JavaScript Variables

When you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

JavaScript For Loop

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

Examples

For loop
How to write a for loop. Use a For loop to run the same block of code a specified number of times.

<html><body> <script type="text/javascript">

for (i = 0; i <= 5; i++){ document.write("The number is " + i); document.write("
"); }

</script> Explanation: This for loop starts with i=0.

As long as i is less than, or equal to 5, the loop will continue to run.

i will increase by 1 each time the loop runs. </body> </html>

Looping through HTML headers
How to use the for loop to loop through the different HTML headers.

<html><body> <script type="text/javascript">

for (i = 1; i <= 6; i++){ document.write("<h" + i + ">This is header " + i);

document.write("</h" + i + ">");} </script> </body> </html>

JavaScript Loops

Very often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

---

The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
   code to be executed
}
```

Example

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 10. **i** will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

Result

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

The while loop

The while loop will be explained in the next chapter.

JavaScript While Loop

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

Examples

While loop
How to write a while loop. Use a while loop to run the same block of code while a specified condition is true.

```
<html><body>  <script type="text/javascript">

i=0;

while (i<=5){ document.write("The number is " + i); document.write("<br />"); i++; }

</script> <p>Explanation:</p>  <p><b>i</b> is equal to 0.</p>

<p>While <b>i</b> is less than , or equal to, 5, the loop will continue to run.</p>

<p><b>i</b> will increase by 1 each time the loop runs.</p> </body> </html>
```

Do while loop
How to write a do...while loop. Use a do...while loop to run the same block of code while a specified condition is true. This loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested.

```
<html><body>

<script type="text/javascript"> i = 0;

do{ document.write("The number is " + i); document.write("<br />"); i++; }

while (i <= 5) </script>  <p>Explanation:</p>  <p><b>i</b>  equal to 0.</p>

<p>The loop will run</p> <p><b>i</b> will increase by 1 each time the loop runs.</p>

<p>While <b>i</b> is less than , or equal to, 5, the loop will continue to run.</p>  </body>

</html>
```

The while loop

The while loop is used when you want the loop to execute and continue executing while the specified condition is true.

```
while (var<=endvalue)
{
   code to be executed
}
```

Note: The <= could be any comparing statement.

Example

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 10. **i** will increase by 1 each time the loop runs.

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=10)
{
document.write("The number is " + i);
document.write("<br />");
i=i+1;
}
</script>
</body>
</html>
```

Result

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will always execute a block of code ONCE, and then it will repeat the loop as long as the specified condition is true. This loop will always be

executed at least once, even if the condition is false, because the code is executed before the condition is tested.

```
do
{
   code to be executed
}
while (var<=endvalue);
```

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
{
document.write("The number is " + i);
document.write("<br />");
i=i+1;
}
while (i<0);
</script>
</body>
</html>
```

Result

```
The number is 0
```

JavaScript Break and Continue

There are two special statements that can be used inside loops: break and continue.

Examples

Break statement
Use the break statement to break the loop.

Continue statement
Use the continue statement to break the current loop and continue with the next value.

JavaScript break and continue Statements

There are two special statements that can be used inside loops: break and continue.

Break

The break command will break the loop and continue executing the code that follows after the loop (if any).

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{
if (i==3)
{
break;
}
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

Result

```
The number is 0
The number is 1
The number is 2
```

Continue

The continue command will break the current loop and continue with the next value.

Example

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
{
if (i==3)
{
continue;
}
document.write("The number is " + i);
document.write("<br />");
}
</script>
```

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

Result

```
The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

JavaScript For...In Statement

The for...in statement is used to loop (iterate) through the elements of an array or through the properties of an object.

Examples

For...In statement
How to use a for...in statement to loop through the elements of an array.

JavaScript For...In Statement

The for...in statement is used to loop (iterate) through the elements of an array or through the properties of an object.

The code in the body of the for ... in loop is executed once for each element/property.

Syntax

```
for (variable in object)
{
   code to be executed
}
```

The variable argument can be a named variable, an array element, or a property of an object.

Example

Using for...in to loop through an array:

```
<html>
<body>
<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";

for (x in mycars)
{
document.write(mycars[x] + "<br />");
}
</script>
</body>
</html>
```

JavaScript Events

Events are actions that can be detected by JavaScript.

Events

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger JavaScript functions. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input box in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

For a complete reference of the events recognized by JavaScript, go to our complete Event reference.

---

onload and onUnload

The onload and onUnload events are triggered when the user enters or leaves the page.

The onload event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

Both the onload and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

---

onFocus, onBlur and onChange

The onFocus, onBlur and onChange events are often used in combination with validation of form fields.

Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30"
id="email" onchange="checkEmail()">
```

---

onSubmit

The onSubmit event is used to validate ALL form fields before submitting it.

Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.htm"
onsubmit="return checkForm()">
```

---

onMouseOver and onMouseOut

onMouseOver and onMouseOut are often used to create "animated" buttons.

Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver event');return false">
<img src="w3schools.gif" width="100" height="30">
</a>
```

JavaScript Try...Catch Statement

The try...catch statement allows you to test a block of code for errors.

Examples

The try...catch statement
How to write a try...catch statement.

The try...catch statement with a confirm box
Another example of how to write a try...catch statement.

JavaScript - Catching Errors

When browsing Web pages on the internet, we all have seen a JavaScript alert box telling us there is a runtime error and asking "Do you wish to debug?". Error message like this may be useful for developers but not for users. When users see errors, they often leave the Web page.

This chapter will teach you how to trap and handle JavaScript error messages, so you don't lose your audience.There are two ways of catching errors in a Web page:

- By using the **try...catch** statement (available in IE5+, Mozilla 1.0, and Netscape 6)
- By using the **onerror** event. This is the old standard solution to catch errors (available since Netscape 3)

Try...Catch Statement

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

Syntax

```
try
{
//Run some code here
}
catch(err)
{
//Handle errors here
}
```

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example 1

The example below contains a script that is supposed to display the message "Welcome guest!" when you click on a button. However, there's a typo in the message() function. alert() is misspelled as adddlert(). A JavaScript error occurs:

```
<html>
<head>
<script type="text/javascript">
function message()
{
adddlert("Welcome guest!");
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()" />
</body>

</html>
```

To take more appropriate action when an error occurs, you can add a try...catch statement.

The example below contains the "Welcome guest!" example rewritten to use the try...catch statement. Since alert() is misspelled, a JavaScript error occurs. However, this time, the catch block catches the error and executes a custom code to handle it. The code displays a custom error message informing the user what happened:

```
<html>
<head>
<script type="text/javascript">
var txt=""
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Error description: " + err.description + "\n\n";
  txt+="Click OK to continue.\n\n";
  alert(txt);
  }
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()" />
</body>

</html>
```

Example 2

The next example uses a confirm box to display a custom message telling users they can click OK to continue viewing the page or click Cancel to go to the homepage. If the confirm method returns false,

the user clicked Cancel, and the code redirects the user. If the confirm method returns true, the code does nothing:

```
<html>
<head>
<script type="text/javascript">
var txt=""
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Click OK to continue viewing this page,\n";
  txt+="or Cancel to return to the home page.\n\n";
  if(!confirm(txt))
    {
    document.location.href="http://www.w3schools.com/";
    }
  }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

The onerror Event

The onerror event will be explained soon, but first you will learn how to use the throw statement to create an exception. The throw statement can be used together with the try...catch statement.

JavaScript Throw Statement

The throw statement allows you to create an exception.

Examples

The throw statement
How to use the throw statement.

The Throw Statement

The throw statement allows you to create an exception. If you use this statement together with the try...catch statement, you can control program flow and generate accurate error messages.

Syntax

```
throw(exception)
```

The exception can be a string, integer, Boolean or an object.

Note that throw is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example 1

The example below determines the value of a variable called x. If the value of x is higher than 10 or lower than 0 we are going to throw an error. The error is then caught by the catch argument and the proper error message is displayed:

```
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","");
try
{
if(x>10)
throw "Err1";
else if(x<0)
throw "Err2";
}
catch(er)
{
if(er=="Err1")
alert("Error! The value is too high");
if(er == "Err2")
alert("Error! The value is too low");
}
</script>
</body>
</html>
```

JavaScript The onerror Event

Using the onerror event is the old standard solution to catch errors in a web page.

Examples

The onerror event (an example with an error)
How to use the onerror event to catch errors in a web page.

The onerror Event

We have just explained how to use the try...catch statement to catch errors in a web page. Now we are going to explain how to use the onerror event for the same purpose.

The onerror event is fired whenever there is a script error in the page.

To use the onerror event, you must create a function to handle the errors. Then you call the function with the onerror event handler. The event handler is called with three arguments: msg (error message), url (the url of the page that caused the error) and line (the line where the error occurred).

Syntax

```
onerror=handleErr
function handleErr(msg,url,l)
{
//Handle the error here
return true or false
}
```

The value returned by onerror determines whether the browser displays a standard error message. If you return false, the browser displays the standard error message in the JavaScript console. If you return true, the browser does not display the standard error message.

Example

The following example shows how to catch the error with the onerror event:

```
<html>
<head>
<script type="text/javascript">
onerror=handleErr;
var txt="";
function handleErr(msg,url,l)
{
txt="There was an error on this page.\n\n";
txt+="Error: " + msg + "\n";
txt+="URL: " + url + "\n";
txt+="Line: " + l + "\n\n";
txt+="Click OK to continue.\n\n";
alert(txt);
return true;
}
function message()
{
adddlert("Welcome guest!");
}
</script>
</head>
```

```
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

JavaScript Special Characters

In JavaScript you can add special characters to a text string by using the backslash sign.

Insert Special Characters

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Look at the following JavaScript code:

```
var txt="We are the so-called "Vikings" from the north.";
document.write(txt);
```

In JavaScript, a string is started and stopped with either single or double quotes. This means that the string above will be chopped to: We are the so-called

To solve this problem, you must place a backslash (\) before each double quote in "Viking". This turns each double quote into a string literal:

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
```

JavaScript will now output the proper text string: We are the so-called "Vikings" from the north.

Here is another example:

```
document.write ("You \& I are singing!");
```

The example above will produce the following output:

```
You & I are singing!
```

The table below lists other special characters that can be added to a text string with the backslash sign:

Code	Outputs
\'	single quote
\"	double quote
\&	Ampersand
\\	Backslash
\n	new line
\r	carriage return
\t	Tab

\b	Backspace
\f	form feed

JavaScript Guidelines

Some other important things to know when scripting with JavaScript.

JavaScript is Case Sensitive

A function named "myfunction" is not the same as "myFunction" and a variable named "myVar" is not the same as "myvar".

JavaScript is case sensitive - therefore watch your capitalization closely when you create or call variables, objects and functions.

White Space

JavaScript ignores extra spaces. You can add white space to your script to make it more readable. The following lines are equivalent:

```
name="Hege";
name = "Hege";
```

Break up a Code Line

You can break up a code line **within a text string** with a backslash. The example below will be displayed properly:

```
document.write("Hello \
World!");
```

However, you cannot break up a code line like this:

```
document.write \
("Hello World!");
```

JavaScript Objects Introduction

JavaScript is an Object Oriented Programming (OOP) language.

An OOP language allows you to define your own objects and make your own variable types.

Object Oriented Programming

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.

However, creating your own objects will be explained later, in the Advanced JavaScript section. We will start by looking at the built-in JavaScript objects, and how they are used. The next pages will explain each built-in JavaScript object in detail.

Note that an object is just a special kind of data. An object has properties and methods.

Properties

Properties are the values associated with an object.

In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">
var txt="Hello World!";
document.write(txt.length);
</script>
```

The output of the code above will be:

```
12
```

Methods

Methods are the actions that can be performed on objects.

In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

The output of the code above will be:

```
HELLO WORLD!
```

JavaScript String Object

The String object is used to manipulate a stored piece of text.

Examples

Return the length of a string
How to use the length property to find the length of a string.

```
<html><body>  <script type="text/javascript">  var txt="Hello World!"; document.write(txt.length);

</script>  </body> </html>
```

Style strings
How to style strings.

```
<html><body>  <script type="text/javascript">

var txt="Hello World!";

document.write("<p>Big: " + txt.big() + "</p>");

document.write("<p>Small: " + txt.small() + "</p>");

document.write("<p>Bold: " + txt.bold() + "</p>");

document.write("<p>Italic: " + txt.italics() + "</p>");

document.write("<p>Blink: " + txt.blink() + " (does not work in IE)</p>");

document.write("<p>Fixed: " + txt.fixed() + "</p>");

document.write("<p>Strike: " + txt.strike() + "</p>");

document.write("<p>Fontcolor: " + txt.fontcolor("Red") + "</p>");

document.write("<p>Fontsize: " + txt.fontsize(16) + "</p>");

document.write("<p>Lowercase: " + txt.toLowerCase() + "</p>");

document.write("<p>Uppercase: " + txt.toUpperCase() + "</p>");

document.write("<p>Subscript: " + txt.sub() + "</p>");

document.write("<p>Superscript: " + txt.sup() + "</p>");

document.write("<p>Link: " + txt.link("http://www.w3schools.com") + "</p>"); </script>

</body></html>
```

The indexOf() method

How to use the indexOf() method to return the position of the first occurrence of a specified string value in a string.

```
<html><body> <script type="text/javascript">

var str="Hello world!"; document.write(str.indexOf("Hello") + "<br />");

document.write(str.indexOf("World") + "<br />"); document.write(str.indexOf("world"));

</script>  </body> </html>
```

The match() method

How to use the match() method to search for a specified string value within a string and return the string value if found

```
<html><body>  <script type="text/javascript">

var str="Hello world!"; document.write(str.match("world") + "<br />");

document.write(str.match("World") + "<br />"); document.write(str.match("worlld") + "<br />");

document.write(str.match("world!")); </script>  </body> </html>
```

Replace characters in a string - replace()

How to use the replace() method to replace some characters with some other characters in a string.

```
<html><body>  <script type="text/javascript">  var str="Visit Microsoft!";

document.write(str.replace(/Microsoft/,"W3Schools")); </script> </body> </html>
```

Complete String Object Reference

For a complete reference of all the properties and methods that can be used with the String object, go to our complete String object reference.

The reference contains a brief description and examples of use for each property and method!

String object

The String object is used to manipulate a stored piece of text.

Examples of use:

The following example uses the length property of the String object to find the length of a string:

```
var txt="Hello world!";
document.write(txt.length);
```

The code above will result in the following output:

```
12
```

The following example uses the toUpperCase() method of the String object to convert a string to uppercase letters:

```
var txt="Hello world!";
document.write(txt.toUpperCase());
```

The code above will result in the following output:

```
HELLO WORLD!
```

JavaScript Date Object

The Date object is used to work with dates and times.

Examples

Return today's date and time
How to use the Date() method to get today's date.

<html><body> <script type="text/javascript"> document.write(Date()); </script> </body>

</html>

getTime()
Use getTime() to calculate the years since 1970.

<html><body> <script type="text/javascript">

var minutes = 1000*60; var hours = minutes*60;var days = hours*24; var years = days*365;

var d = new Date();var t = d.getTime(); var y = t/years;

document.write("It's been: " + y + " years since 1970/01/01!"); </script> </body> </html>

setFullYear()
How to use setFullYear() to set a specific date.

<html><body> <script type="text/javascript"> var d = new Date(); d.setFullYear(1992,10,3);

document.write(d); </script> </body> </html>

toUTCString()
How to use toUTCString() to convert today's date (according to UTC) to a string.

<html><body> <script type="text/javascript">

var d = new Date();document.write (d.toUTCString()); </script> </body> </html>

getDay()
Use getDay() and an array to write a weekday, and not just a number.

```
<html><body>  <script type="text/javascript">  var d=new Date(); var weekday=new Array(7);

weekday[0]="Sunday";weekday[1]="Monday"; weekday[2]="Tuesday"; weekday[3]="Wednesday";

weekday[4]="Thursday";weekday[5]="Friday"; weekday[6]="Saturday";

document.write("Today it is " + weekday[d.getDay()]); </script>  </body> </html>
```

Display a clock
How to display a clock on your web page.

```
<html><head> <script type="text/javascript">

function startTime() { var today=new Date(); var h=today.getHours(); var m=today.getMinutes();

var s=today.getSeconds(); // add a zero in front of numbers<10

m=checkTime(m); s=checkTime(s);

document.getElementById('txt').innerHTML=h+":"+m+":"+s; t=setTimeout('startTime()',500); }

function checkTime(i){

if (i<10)  {   i="0" + i;   } return i; } </script> </head>

<body onload="startTime()"> <div id="txt"></div> </body> </html>
```

Complete Date Object Reference

For a complete reference of all the properties and methods that can be used with the Date object, go to our complete Date object reference.

The reference contains a brief description and examples of use for each property and method!

Create a Date Object

The Date object is used to work with dates and times.

The following code create a Date object called myDate:

```
var myDate=new Date()
```

Note: The Date object will automatically hold the current date and time as its initial value!

Set Dates

We can easily manipulate the date by using the methods available for the Date object.

In the example below we set a Date object to a specific date (14th January 2010):

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
```

And in the following example we set a Date object to be 5 days into the future:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+5);
```

Note: If adding five days to a date shifts the month or year, the changes are handled automatically by the Date object itself!

Compare Two Dates

The Date object is also used to compare two dates.

The following example compares today's date with the 14th January 2010:

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
var today = new Date();
if (myDate>today)
{
alert("Today is before 14th January 2010");
}
else
{
alert("Today is after 14th January 2010");
}
```

JavaScript Array Object

The Array object is used to store multiple values in a single variable.

Examples

Create an array
Create an array, assign values to it, and write the values to the output.

<html> <body> <script type="text/javascript"> var mycars = new Array();

```
mycars[0] = "Saab";mycars[1] = "Volvo"; mycars[2] = "BMW";

for (i=0;i<mycars.length;i++){ document.write(mycars[i] + "<br />"); } </script>  </body>

</html>
```

For...In Statement

How to use a for...in statement to loop through the elements of an array.

```
<html><body> <script type="text/javascript"> var x; var mycars = new Array();

mycars[0] = "Saab";mycars[1] = "Volvo"; mycars[2] = "BMW";

for (x in mycars){ document.write(mycars[x] + "<br />"); } </script> </body> </html>
```

Join two arrays - concat()

How to use the concat() method to join two arrays.

```
<html><body>  <script type="text/javascript">  var arr = new Array(3);

arr[0] = "Jani";arr[1] = "Tove"; arr[2] = "Hege"; var arr2 = new Array(3); arr2[0] = "John";

arr2[1] = "Andy";arr2[2] = "Wendy"; document.write(arr.concat(arr2));  </script>  </body> </html>
```

Put array elements into a string - join()

How to use the join() method to put all the elements of an array into a string.

```
<html><body>  <script type="text/javascript">  var arr = new Array(3);

arr[0] = "Jani";arr[1] = "Hege"; arr[2] = "Stale"; document.write(arr.join() + "<br />");

document.write(arr.join(".")); </script>  </body> </html>
```

Literal array - sort()

How to use the sort() method to sort a literal array.

```
<html><body>  <script type="text/javascript">  var arr = new Array(6);

arr[0] = "Jani";arr[1] = "Hege"; arr[2] = "Stale"; arr[3] = "Kai Jim"; arr[4] = "Borge";

arr[5] = "Tove";

document.write(arr + "<br />"); document.write(arr.sort()); </script>  </body> </html>
```

Numeric array - sort()

How to use the sort() method to sort a numeric array.

```
<html> <body>  <script type="text/javascript">

function sortNumber(a, b){ return a - b; }

var arr = new Array(6);arr[0] = "10"; arr[1] = "5"; arr[2] = "40"; arr[3] = "25"; arr[4] = "1000";
```

```
arr[5] = "1";

document.write(arr + "<br />"); document.write(arr.sort(sortNumber)); </script>  </body> </html>
```

Complete Array Object Reference

For a complete reference of all the properties and methods that can be used with the Array object, go to our complete Array object reference.

The reference contains a brief description and examples of use for each property and method!

Create an Array

The following code creates an Array object called myCars:

```
var myCars=new Array()
```

There are two ways of adding values to an array (you can add as many values as you need to define as many variables you require).

1:

```
var myCars=new Array();
mycars[0]="Saab";
mycars[1]="Volvo";
mycars[2]="BMW";
```

You could also pass an integer argument to control the array's size:

```
var myCars=new Array(3);
mycars[0]="Saab";
mycars[1]="Volvo";
mycars[2]="BMW";
```

2:

```
var myCars=new Array("Saab","Volvo","BMW");
```

Note: If you specify numbers or true/false values inside the array then the type of variables will be numeric or Boolean instead of string.

Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

The following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Saab
```

Modify Values in an Array

To modify a value in an existing array, just add a new value to the array with a specified index number:

```
myCars[0]="Opel";
```

Now, the following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Opel
```

JavaScript Boolean Object

The Boolean object is used to convert a non-Boolean value to a Boolean value (true or false).

Examples

Check Boolean value
Check if a Boolean object is true or false.

<html><body> <script type="text/javascript"> var b1=new Boolean(0);

var b2=new Boolean(1);var b3=new Boolean(""); var b4=new Boolean(null);

var b5=new Boolean(NaN);var b6=new Boolean("false");

document.write("0 is boolean "+ b1 +"
");

document.write("1 is boolean "+ b2 +"
");

document.write("An empty string is boolean "+ b3 + "
");

document.write("null is boolean "+ b4+ "
");

```
document.write("NaN is boolean "+ b5 +"<br />");

document.write("The string 'false' is boolean "+ b6 +"<br />"); </script>  </body> </html>
```

Complete Boolean Object Reference

For a complete reference of all the properties and methods that can be used with the Boolean object, go to our complete Boolean object reference.

The reference contains a brief description and examples of use for each property and method!

Create a Boolean Object

The Boolean object represents two values: "true" or "false".

The following code creates a Boolean object called myBoolean:

```
var myBoolean=new Boolean();
```

Note: If the Boolean object has no initial value or if it is 0, -0, null, "", false, undefined, or NaN, the object is set to false. Otherwise it is true (even with the string "false")!

All the following lines of code create Boolean objects with an initial value of false:

```
var myBoolean=new Boolean();
var myBoolean=new Boolean(0);
var myBoolean=new Boolean(null);
var myBoolean=new Boolean("");
var myBoolean=new Boolean(false);
var myBoolean=new Boolean(NaN);
```

And all the following lines of code create Boolean objects with an initial value of true:

```
var myBoolean=new Boolean(true);
var myBoolean=new Boolean("true");
var myBoolean=new Boolean("false");
var myBoolean=new Boolean("Richard");
```

JavaScript Math Object

The Math object allows you to perform mathematical tasks.

Examples

round()
How to use round().

```
<html><body><script type="text/javascript">

document.write(Math.round(0.60) + "<br />"); document.write(Math.round(0.50) + "<br />");

document.write(Math.round(0.49) + "<br />"); document.write(Math.round(-4.40) + "<br />");

document.write(Math.round(-4.60)); </script>  </body> </html>
```

random()
How to use random() to return a random number between 0 and 1.

```
<html> <body>  <script type="text/javascript">  document.write(Math.random());  </script>

</body> </html>
```

max()
How to use max() to return the number with the highest value of two specified numbers.

```
<html> <body>  <script type="text/javascript">

document.write(Math.max(5,7) + "<br />"); document.write(Math.max(-3,5) + "<br />");

document.write(Math.max(-3,-5) + "<br />"); document.write(Math.max(7.25,7.30));

</script>  </body> </html>
```

min()
How to use min() to return the number with the lowest value of two specified numbers.

```
<html> <body>  <script type="text/javascript">

document.write(Math.min(5,7) + "<br />"); document.write(Math.min(-3,5) + "<br />");

document.write(Math.min(-3,-5) + "<br />"); document.write(Math.min(7.25,7.30));

</script>  </body> </html>
```

Complete Math Object Reference

For a complete reference of all the properties and methods that can be used with the Math object, go to our complete Math object reference.

The reference contains a brief description and examples of use for each property and method!

Math Object

The Math object allows you to perform mathematical tasks.

The Math object includes several mathematical constants and methods.

Syntax for using properties/methods of Math:

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

Note: Math is not a constructor. All properties and methods of Math can be called by using Math as an object without creating it.

Mathematical Constants

JavaScript provides eight mathematical constants that can be accessed from the Math object. These are: E, PI, square root of 2, square root of 1/2, natural log of 2, natural log of 10, base-2 log of E, and base-10 log of E.

You may reference these constants from your JavaScript like this:

```
Math.E
Math.PI
Math.SQRT2
Math.SQRT1_2
Math.LN2
Math.LN10
Math.LOG2E
Math.LOG10E
```

Mathematical Methods

In addition to the mathematical constants that can be accessed from the Math object there are also several methods available.

The following example uses the round() method of the Math object to round a number to the nearest integer:

```
document.write(Math.round(4.7));
```

The code above will result in the following output:

```
5
```

The following example uses the random() method of the Math object to return a random number between 0 and 1:

```
document.write(Math.random());
```

The code above can result in the following output:

```
0.503799862270377
```

The following example uses the floor() and random() methods of the Math object to return a random number between 0 and 10:

```
document.write(Math.floor(Math.random()*11));
```

The code above can result in the following output:

```
7
```

JavaScript RegExp Object

The RegExp object is used to specify what to search for in a text

What is RegExp

RegExp, is short for regular expression.

When you search in a text, you can use a pattern to describe what you are searching for. **RegExp IS this pattern**.

A simple pattern can be a single character.

A more complicated pattern consists of more characters, and can be used for parsing, format checking, substitution and more.

You can specify where in the string to search, what type of characters to search for, and more.

Defining RegExp

The RegExp object is used to store the search pattern.

We define a RegExp object with the *new* keyword. The following code line defines a RegExp object called patt1 with the pattern "e":

```
var patt1=new RegExp("e");
```

When you use this RegExp object to search in a string, you will find the letter "e".

Methods of the RegExp Object

The RegExp Object has 3 methods: test(), exec(), and compile().

test()

The test() method searches a string for a specified value. Returns true or false

Example:

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
true
```

---

exec()

The exec() method searches a string for a specified value. Returns the text of the found value. If no match is found, it returns *null*

Example 1:

```
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
e
```

Example 2:

You can add a second parameter to the RegExp object, to specify your search. For example; if you want to find all occurrences of a character, you can use the "g" parameter ("global").

For a complete list of how to modify your search, visit our complete RegExp object reference.

When using the "g" parameter, the exec() method works like this:

- Finds the first occurence of "e", and stores its position
- If you run exec() again, it starts at the stored position, and finds the next occurence of "e", and stores its position

```
var patt1=new RegExp("e","g");
do
{
result=patt1.exec("The best things in life are free");
document.write(result);
}
while (result!=null)
```

Since there is six "e" letters in the string, the output of the code above will be:

```
eeeeeenull
```

---

compile()

The compile() method is used to change the RegExp.

compile() can change both the search pattern, and add or remove the second parameter.

Example:

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
patt1.compile("d");
document.write(patt1.test("The best things in life are free"));
```

Since there is an "e" in the string, but not a "d", the output of the code above will be:

```
truefalse
```

Complete RegExp Object Reference

For a complete reference of all the properties and methods that can be used with the RegExp object, go to our complete RegExp object reference.

The reference contains a brief description and examples of use for each property and method including the string object

JavaScript HTML DOM Objects

In addition to the built-in JavaScript objects, you can also access and manipulate all of the HTML DOM objects with JavaScript.

More JavaScript Objects

Follow the links to learn more about the objects and their collections, properties, methods and events.

Object	Description
Window	The top level object in the JavaScript hierarchy. The Window object represents a browser window. A Window object is created automatically with every instance of a <body> or <frameset> tag
Navigator	Contains information about the client's browser
Screen	Contains information about the client's display screen
History	Contains the visited URLs in the browser window
Location	Contains information about the current URL

The HTML DOM

The HTML DOM is a W3C standard and it is an abbreviation for the Document Object Model for HTML.

The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML documents.

All HTML elements, along with their containing text and attributes, can be accessed through the DOM. The contents can be modified or deleted, and new elements can be created.

The HTML DOM is platform and language independent. It can be used by any programming language like Java, JavaScript, and VBScript.Follow the links below to learn more about how to access and manipulate each DOM object with JavaScript:

Object	Description
Document	Represents the entire HTML document and can be used to access all elements in a page
Anchor	Represents an <a> element
Area	Represents an <area> element inside an image-map
Base	Represents a <base> element
Body	Represents the <body> element
Button	Represents a <button> element
Event	Represents the state of an event
Form	Represents a <form> element
Frame	Represents a <frame> element
Frameset	Represents a <frameset> element
Iframe	Represents an <iframe> element
Image	Represents an <img> element
Input button	Represents a button in an HTML form
Input checkbox	Represents a checkbox in an HTML form
Input file	Represents a fileupload in an HTML form
Input hidden	Represents a hidden field in an HTML form
Input password	Represents a password field in an HTML form
Input radio	Represents a radio button in an HTML form
Input reset	Represents a reset button in an HTML form
Input submit	Represents a submit button in an HTML form
Input text	Represents a text-input field in an HTML form
Link	Represents a <link> element
Meta	Represents a <meta> element
Option	Represents an <option> element
Select	Represents a selection list in an HTML form
Style	Represents an individual style statement
Table	Represents a <table> element
TableData	Represents a <td> element
TableRow	Represents a <tr> element
Textarea	Represents a <textarea> element

JavaScript Browser Detection

The JavaScript Navigator object contains information about the visitor's browser.

Examples

Detect the visitor's browser and browser version

```
<html> <body> <script type="text/javascript"> var browser=navigator.appName;
var b_version=navigator.appVersion; var version=parseFloat(b_version);
document.write("Browser name: "+ browser); document.write("<br />");
document.write("Browser version: "+ version); </script> </body> </html>
```

More details about the visitor's browser

```
<html> <body> <script type="text/javascript"> document.write("<p>Browser: ");
document.write(navigator.appName + "</p>");  document.write("<p>Browserversion: ");
document.write(navigator.appVersion + "</p>"); document.write("<p>Code: ");
document.write(navigator.appCodeName + "</p>"); document.write("<p>Platform: ");
document.write(navigator.platform + "</p>");  document.write("<p>Cookies enabled: ");
document.write(navigator.cookieEnabled + "</p>");
document.write("<p>Browser's user agent header: ");
document.write(navigator.userAgent + "</p>"); </script> </body> </html>
```

All details about the visitor's browser

```
<html> <body>  <script type="text/javascript"> var x = navigator;
document.write("CodeName=" + x.appCodeName); document.write("<br />");
document.write("MinorVersion=" + x.appMinorVersion); document.write("<br />");
document.write("Name=" + x.appName); document.write("<br />");
document.write("Version=" + x.appVersion); document.write("<br />");
document.write("CookieEnabled=" + x.cookieEnabled); document.write("<br />");
document.write("CPUClass=" + x.cpuClass); document.write("<br />");
```

```
document.write("OnLine=" + x.onLine); document.write("<br />");

document.write("Platform=" + x.platform); document.write("<br />");

document.write("UA=" + x.userAgent); document.write("<br />");

document.write("BrowserLanguage=" + x.browserLanguage); document.write("<br />");

document.write("SystemLanguage=" + x.systemLanguage); document.write("<br />");

document.write("UserLanguage=" + x.userLanguage); </script> </body> </html>
```

Alert user, depending on browser

```
<html><head> <script type="text/javascript">

function detectBrowser() { var browser=navigator.appName; var b_version=navigator.appVersion;

var version=parseFloat(b_version);

if ((browser=="Netscape"||browser=="Microsoft Internet Explorer") && (version>=4))

 {   alert("Your browser is good enough!");   }

Else   {   alert("It's time to upgrade your browser!");   } }

</script> </head>

<body onload="detectBrowser()"> </body>  </html>
```

Browser Detection

Almost everything in this tutorial works on all JavaScript-enabled browsers. However, there are some things that just don't work on certain browsers - especially on older browsers.

So, sometimes it can be very useful to detect the visitor's browser type and version, and then serve up the appropriate information.

The best way to do this is to make your web pages smart enough to look one way to some browsers and another way to other browsers.

JavaScript includes an object called the Navigator object, that can be used for this purpose.

The Navigator object contains information about the visitor's browser name, browser version, and more.

The Navigator Object

The JavaScript Navigator object contains all information about the visitor's browser. We are going to look at two properties of the Navigator object:

- appName - holds the name of the browser
- appVersion - holds, among other things, the version of the browser

Example

```
<html>
<body>
<script type="text/javascript">
var browser=navigator.appName;
var b_version=navigator.appVersion;
var version=parseFloat(b_version);
document.write("Browser name: "+ browser);
document.write("<br />");
document.write("Browser version: "+ version);
</script>
</body>
</html>
```

The variable browser in the example above holds the name of the browser, i.e. "Netscape" or "Microsoft Internet Explorer".

The appVersion property in the example above returns a string that contains much more information than just the version number, but for now we are only interested in the version number. To pull the version number out of the string we are using a function called parseFloat(), which pulls the first thing that looks like a decimal number out of a string and returns it.

IMPORTANT! The version number is WRONG in IE 5.0 or later! Microsoft starts the appVersion string with the number 4.0. in IE 5.0 and IE 6.0!!! Why did they do that??? However, JavaScript is the same in IE6, IE5 and IE4, so for most scripts it is ok.

Example

The script below displays a different alert, depending on the visitor's browser:

```
<html>
<head>
<script type="text/javascript">
function detectBrowser()
{
var browser=navigator.appName;
var b_version=navigator.appVersion;
var version=parseFloat(b_version);
if ((browser=="Netscape"||browser=="Microsoft Internet Explorer")
&& (version>=4))
{
alert("Your browser is good enough!");
}
else
{
alert("It's time to upgrade your browser!");
```

```
}
}
</script>
</head>
<body onload="detectBrowser()">
</body>
</html>
```

JavaScript Cookies

A cookie is often used to identify a user.

Examples

Create a welcome cookie

<html> <head> <script type="text/javascript">

function getCookie(c_name) { if (document.cookie.length>0) {

c_start=document.cookie.indexOf(c_name + "=");

if (c_start!=-1) { c_start=c_start + c_name.length+1 ;

c_end=document.cookie.indexOf(";",c_start);

if (c_end==-1) c_end=document.cookie.length

return unescape(document.cookie.substring(c_start,c_end)); } } return "" }

function setCookie(c_name,value,expiredays) { var exdate=new Date();

exdate.setDate(exdate.getDate()+expiredays);

document.cookie=c_name+ "=" +escape(value)+((expiredays==null) ? "" : ";
expires="+exdate.toGMTString()); }

function checkCookie() { username=getCookie('username');

if (username!=null && username!="") { alert('Welcome again '+username+'!'); }

else { username=prompt('Please enter your name:',"");

if (username!=null && username!="") { setCookie('username',username,365); } } }
</script> </head> <body onLoad="checkCookie()"> </body> </html>

What is a Cookie?

A cookie is a variable that is stored on the visitor's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With JavaScript, you can both create and retrieve cookie values.

Examples of cookies:

- Name cookie - The first time a visitor arrives to your web page, he or she must fill in her/his name. The name is then stored in a cookie. Next time the visitor arrives at your page, he or she could get a welcome message like "Welcome John Doe!" The name is retrieved from the stored cookie
- Password cookie - The first time a visitor arrives to your web page, he or she must fill in a password. The password is then stored in a cookie. Next time the visitor arrives at your page, the password is retrieved from the cookie
- Date cookie - The first time a visitor arrives to your web page, the current date is stored in a cookie. Next time the visitor arrives at your page, he or she could get a message like "Your last visit was on Tuesday August 11, 2005!" The date is retrieved from the stored cookie

---

Create and Store a Cookie

In this example we will create a cookie that stores the name of a visitor. The first time a visitor arrives to the web page, he or she will be asked to fill in her/his name. The name is then stored in a cookie. The next time the visitor arrives at the same page, he or she will get welcome message.

First, we create a function that stores the name of the visitor in a cookie variable:

```
function setCookie(c_name,value,expiredays)
{
var exdate=new Date();
exdate.setDate(exdate.getDate()+expiredays);
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
}
```

The parameters of the function above hold the name of the cookie, the value of the cookie, and the number of days until the cookie expires.

In the function above we first convert the number of days to a valid date, then we add the number of days until the cookie should expire. After that we store the cookie name, cookie value and the expiration date in the document.cookie object.

Then, we create another function that checks if the cookie has been set:

```
function getCookie(c_name)
{
if (document.cookie.length>0)
  {
  c_start=document.cookie.indexOf(c_name + "=");
  if (c_start!=-1)
    {
    c_start=c_start + c_name.length+1;
```

```
    c_end=document.cookie.indexOf(";",c_start);
    if (c_end==-1) c_end=document.cookie.length;
    return unescape(document.cookie.substring(c_start,c_end));
    }
  }
return "";
}
```

The function above first checks if a cookie is stored at all in the document.cookie object. If the document.cookie object holds some cookies, then check to see if our specific cookie is stored. If our cookie is found, then return the value, if not - return an empty string.

Last, we create the function that displays a welcome message if the cookie is set, and if the cookie is not set it will display a prompt box, asking for the name of the user:

```
function checkCookie()
{
username=getCookie('username');
if (username!=null && username!="")
{
alert('Welcome again '+username+'!');
}
else
{
username=prompt('Please enter your name:',"");
if (username!=null && username!="")
{
setCookie('username',username,365);
}
}
}
```

All together now:

```
<html>
<head>
<script type="text/javascript">
function getCookie(c_name)
{
if (document.cookie.length>0)
  {
  c_start=document.cookie.indexOf(c_name + "=");
  if (c_start!=-1)
    {
    c_start=c_start + c_name.length+1;
    c_end=document.cookie.indexOf(";",c_start);
    if (c_end==-1) c_end=document.cookie.length;
    return unescape(document.cookie.substring(c_start,c_end));
    }
  }
return "";
}
function setCookie(c_name,value,expiredays)
{
var exdate=new Date();
```

```
exdate.setDate(exdate.getDate()+expiredays);
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
}
function checkCookie()
{
username=getCookie('username');
if (username!=null && username!="")
  {
  alert('Welcome again '+username+'!');
  }
  else
  {
  username=prompt('Please enter your name:',"");
  if (username!=null && username!="")
    {
    setCookie('username',username,365);
    }
  }
}
</script>
</head>
<body onLoad="checkCookie()">
</body>
</html>
```

The example above runs the checkCookie() function when the page loads.

JavaScript Form Validation

JavaScript can be used to validate input data in HTML forms before sending off the content to a server.

JavaScript Form Validation

JavaScript can be used to validate input data in HTML forms before sending off the content to a server.

Form data that typically are checked by a JavaScript could be:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Required Fields

The function below checks if a required field has been left empty. If the required field is blank, an alert box alerts a message and the function returns false. If a value is entered, the function returns true (means that data is OK):

```
function validate_required(field,alerttxt)
{
with (field)
{
  if (value==null||value=="")
  {
  alert(alerttxt);return false;
  }
  else
  {
  return true;
  }
}
}
```

The entire script, with the HTML form could look something like this:

```
<html>
<head>
<script type="text/javascript">
function validate_required(field,alerttxt)
{
with (field)
{
if (value==null||value=="")
  {alert(alerttxt);return false;}
else {return true}
}
}
function validate_form(thisform)
{
with (thisform)
{
if (validate_required(email,"Email must be filled out!")==false)
  {email.focus();return false;}
}
}
</script>
</head>
<body>
<form action="submitpage.htm"
onsubmit="return validate_form(this)"
method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

E-mail Validation

The function below checks if the content has the general syntax of an email.

This means that the input data must contain at least an @ sign and a dot (.). Also, the @ must not be the first character of the email address, and the last dot must at least be one character after the @ sign:

```
function validate_email(field,alerttxt)
{
with (field)
{
apos=value.indexOf("@");
dotpos=value.lastIndexOf(".");
if (apos<1||dotpos-apos<2)
  {alert(alerttxt);return false;}
else {return true;}
}
}
```

The entire script, with the HTML form could look something like this:

```
<html>
<head>
<script type="text/javascript">
function validate_email(field,alerttxt)
{
with (field)
{
apos=value.indexOf("@");
dotpos=value.lastIndexOf(".");
if (apos<1||dotpos-apos<2)
  {alert(alerttxt);return false;}
else {return true;}
}
}
function validate_form(thisform)
{
with (thisform)
{
if (validate_email(email,"Not a valid e-mail address!")==false)
  {email.focus();return false;}
}
}
</script>
</head>
<body>
<form action="submitpage.htm"
onsubmit="return validate_form(this);"
method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

JavaScript Animation

With JavaScript we can create animated images.

Examples

Button animation

<html> <head> <script type="text/javascript">

function mouseOver() { document.b1.src ="b_blue.gif"; }

function mouseOut() { document.b1.src ="b_pink.gif"; } </script> </head>

<body>

 </body> </html>

JavaScript Animation

It is possible to use JavaScript to create animated images.

The trick is to let a JavaScript change between different images on different events.

In the following example we will add an image that should act as a link button on a web page. We will then add an onMouseOver event and an onMouseOut event that will run two JavaScript functions that will change between the images.

The HTML Code

The HTML code looks like this:

```
<a href="http://www.w3schools.com" target="_blank">
<img border="0" alt="Visit W3Schools!"
src="b_pink.gif" name="b1"
onmouseOver="mouseOver()"
onmouseOut="mouseOut()" />
</a>
```

Note that we have given the image a name to make it possible for JavaScript to address it later.

The onMouseOver event tells the browser that once a mouse is rolled over the image, the browser should execute a function that will replace the image with another image.

The onMouseOut event tells the browser that once a mouse is rolled away from the image, another JavaScript function should be executed. This function will insert the original image again.

The JavaScript Code

The changing between the images is done with the following JavaScript:

```
<script type="text/javascript">
function mouseOver()
{
document.b1.src ="b_blue.gif";
}
function mouseOut()
{
document.b1.src ="b_pink.gif";
}
</script>
```

The function mouseOver() causes the image to shift to "b_blue.gif".

The function mouseOut() causes the image to shift to "b_pink.gif".

The Entire Code

```
<html>
<head>
<script type="text/javascript">
function mouseOver()
{
document.b1.src ="b_blue.gif";
}
function mouseOut()
{
document.b1.src ="b_pink.gif";
}
</script>
</head>

<body>
<a href="http://www.w3schools.com" target="_blank">
<img border="0" alt="Visit W3Schools!"
src="b_pink.gif" name="b1"
onmouseOver="mouseOver()"
onmouseOut="mouseOut()" />
</a>
</body>
</html>
```

JavaScript Image Maps

An image-map is an image with clickable regions.

Examples

Simple HTML Image map

Image map with added JavaScript

HTML Image Maps

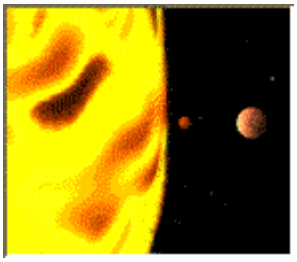
From our HTML tutorial we have learned that an image-map is an image with clickable regions. Normally, each region has an associated hyperlink. Clicking on one of the regions takes you to the associated link.

Example

The example below demonstrates how to create an HTML image map, with clickable regions. Each of the regions is a hyperlink:

```
<img src ="planets.gif"
width ="145" height ="126"
alt="Planets"
usemap ="#planetmap" />
<map id ="planetmap"
name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
  href ="sun.htm" target ="_blank"
  alt="Sun" />
<area shape ="circle" coords ="90,58,3"
  href ="mercur.htm" target ="_blank"
  alt="Mercury" />
<area shape ="circle" coords ="124,58,8"
  href ="venus.htm" target ="_blank"
  alt="Venus" />
</map>
```

Result



Adding some JavaScript

We can add events (that can call a JavaScript) to the <area> tags inside the image map. The <area> tag supports the onClick, onDblClick, onMouseDown, onMouseUp, onMouseOver, onMouseMove, onMouseOut, onKeyPress, onKeyDown, onKeyUp, onFocus, and onBlur events.

Here's the above example, with some JavaScript added:

```
<html>
<head>
<script type="text/javascript">
function writeText(txt)
{
document.getElementById("desc").innerHTML=txt;
}
</script>
</head>
<body>
<img src="planets.gif" width="145" height="126"
alt="Planets" usemap="#planetmap" />

<map id ="planetmap" name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
onMouseOver="writeText('The Sun and the gas giant
planets like Jupiter are by far the largest objects
in our Solar System.')"
href ="sun.htm" target ="_blank" alt="Sun" />

<area shape ="circle" coords ="90,58,3"
onMouseOver="writeText('The planet Mercury is very
difficult to study from the Earth because it is
always so close to the Sun.')"
href ="mercur.htm" target ="_blank" alt="Mercury" />

<area shape ="circle" coords ="124,58,8"
onMouseOver="writeText('Until the 1960s, Venus was
often considered a twin sister to the Earth because
Venus is the nearest planet to us, and because the
two planets seem to share many characteristics.')"
href ="venus.htm" target ="_blank" alt="Venus" />
</map>

<p id="desc"></p>

</body>
</html>
```

JavaScript Timing Events

With JavaScript, it is possible to execute some code NOT immediately after a function is called, but after a specified time interval. This is called timing events.

Examples

Simple timing

```
<html> <head> <script type="text/javascript">

function timedMsg() { var t=setTimeout("alert('5 seconds!')",5000); } </script> </head>

<body> <form> <input type="button" value="Display timed alertbox!" onClick = "timedMsg()">

</form> <p>Click on the button above. An alert box will be displayed after 5 seconds.</p> </body>

</html>
```

Another simple timing

```
<html> <head> <script type="text/javascript">

function timedText() {

var t1=setTimeout("document.getElementById('txt').value='2 seconds!'",2000);

var t2=setTimeout("document.getElementById('txt').value='4 seconds!'",4000);

var t3=setTimeout("document.getElementById('txt').value='6 seconds!'",6000); } </script> </head>

<body> <form> <input type="button" value="Display timed text!" onClick="timedText()">

<input type="text" id="txt"> </form>

<p>Click on the button above. The input field will tell you when two, four, and six seconds have
passed.</p> </body>  </html>
```

Timing event in an infinite loop

```
<html> <head> <script type="text/javascript"> var c=0; var t;

function timedCount() { document.getElementById('txt').value=c;
c=c+1;t=setTimeout("timedCount()",1000); } </script> </head>   <body> <form>

<input type="button" value="Start count!" onClick="timedCount()"> <input type="text" id="txt">

</form> <p>Click on the button above. The input field will count for ever, starting at 0.</p>

</body>  </html>
```

Timing event in an infinite loop - with a Stop button

```
<html> <head> <script type="text/javascript"> var c=0; var t;

function timedCount() { document.getElementById('txt').value=c; c=c+1;
```

t=setTimeout("timedCount()",1000); }

function stopCount() { clearTimeout(t); } </script> </head>

<body> <form><input type="button" value="Start count!" onClick="timedCount()">

<input type="text" id="txt"> <input type="button" value="Stop count!" onClick="stopCount()">

</form> Click on the "Start count!" button above to start the timer. The input field will count forever, starting at 0. Click on the "Stop count!" button to stop the counting. </body> </html>

A clock created with a timing event

<html> <head> <script type="text/javascript">

function startTime() { var today=new Date(); var h=today.getHours(); var m=today.getMinutes();

var s=today.getSeconds(); // add a zero in front of numbers<10

m=checkTime(m); s=checkTime(s); document.getElementById('txt').innerHTML=h+":"+m+":"+s;

t=setTimeout('startTime()',500); } function checkTime(i)

{ if (i<10) { i="0" + i; } return i; } </script> </head>

<body onload="startTime()"> </body> </html>

JavaScript Timing Events

With JavaScript, it is possible to execute some code NOT immediately after a function is called, but after a specified time interval. This is called timing events.

It's very easy to time events in JavaScript. The two key methods that are used are:

- setTimeout() - executes a code some time in the future
- clearTimeout() - cancels the setTimeout()

Note: The setTimeout() and clearTimeout() are both methods of the HTML DOM Window object.

---

setTimeout()

Syntax

```
var t=setTimeout("javascript statement",milliseconds);
```

The setTimeout() method returns a value - In the statement above, the value is stored in a variable called t. If you want to cancel this setTimeout(), you can refer to it using the variable name.

The first parameter of setTimeout() is a string that contains a JavaScript statement. This statement could be a statement like "alert('5 seconds!')" or a call to a function, like "alertMsg()".

The second parameter indicates how many milliseconds from now you want to execute the first parameter.

Note: There are 1000 milliseconds in one second.

Example

When the button is clicked in the example below, an alert box will be displayed after 5 seconds.

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
var t=setTimeout("alert('5 seconds!')",5000);
}
</script>
</head>
<body>
<form>
<input type="button" value="Display timed alertbox!"
onClick="timedMsg()">
</form>
</body>
</html>
```

Example - Infinite Loop

To get a timer to work in an infinite loop, we must write a function that calls itself. In the example below, when the button is clicked, the input field will start to count (for ever), starting at 0:

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c;
c=c+1;
t=setTimeout("timedCount()",1000);
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!"
onClick="timedCount()">
<input type="text" id="txt">
</form>
</body>
</html>
```

clearTimeout()

Syntax

```
clearTimeout(setTimeout_variable)
```

Example

The example below is the same as the "Infinite Loop" example above. The only difference is that we have now added a "Stop Count!" button that stops the timer:

```
<html>
<head>
<script type="text/javascript">
var c=0
var t
function timedCount()
{
document.getElementById('txt').value=c;
c=c+1;
t=setTimeout("timedCount()",1000);
}
function stopCount()
{
clearTimeout(t);
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!"
onClick="timedCount()">
<input type="text" id="txt">
<input type="button" value="Stop count!"
onClick="stopCount()">
</form>
</body>
</html>
```

JavaScript Create Your Own Objects

Objects are useful to organize information.

Examples

Create a direct instance of an object

<html> <body> <script type="text/javascript"> personObj=new Object();

personObj.firstname="John"; personObj.lastname="Doe"; personObj.age=50;

personObj.eyecolor="blue";

document.write(personObj.firstname + " is " + personObj.age + " years old."); </script> </body>

</html>

Create a template for an object

<html> <body> <script type="text/javascript">

function person(firstname,lastname,age,eyecolor) { this.firstname=firstname;

this.lastname=lastname; this.age=age; this.eyecolor=eyecolor; }

myFather=new person("John","Doe",50,"blue");

document.write(myFather.firstname + " is " + myFather.age + " years old."); </script>

</body> </html>

JavaScript Objects

Earlier in this tutorial we have seen that JavaScript has several built-in objects, like String, Date, Array, and more. In addition to these built-in objects, you can also create your own.

An object is just a special kind of data, with a collection of properties and methods.

Let's illustrate with an example: A person is an object. Properties are the values associated with the object. The persons' properties include name, height, weight, age, skin tone, eye color, etc. All persons have these properties, but the values of those properties will differ from person to person. Objects also have methods. Methods are the actions that can be performed on objects. The persons' methods could be eat(), sleep(), work(), play(), etc.

Properties

The syntax for accessing a property of an object is:

```
objName.propName
```

You can add properties to an object by simply giving it a value. Assume that the personObj already exists - you can give it properties named firstname, lastname, age, and eyecolor as follows:

```
personObj.firstname="John";
personObj.lastname="Doe";
personObj.age=30;
personObj.eyecolor="blue";
document.write(personObj.firstname);
```

The code above will generate the following output:

```
John
```

Methods

An object can also contain methods.

You can call a method with the following syntax:

```
objName.methodName()
```

Note: Parameters required for the method can be passed between the parentheses.

To call a method called sleep() for the personObj:

```
personObj.sleep();
```

Creating Your Own Objects

There are different ways to create a new object:

1. Create a direct instance of an object

The following code creates an instance of an object and adds four properties to it:

```
personObj=new Object();
personObj.firstname="John";
personObj.lastname="Doe";
personObj.age=50;
personObj.eyecolor="blue";
```

Adding a method to the personObj is also simple. The following code adds a method called eat() to the personObj:

```
personObj.eat=eat;
```

2. Create a template of an object

The template defines the structure of an object:

```
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname;
this.lastname=lastname;
this.age=age;
this.eyecolor=eyecolor;
}
```

Notice that the template is just a function. Inside the function you need to assign things to this.propertyName. The reason for all the "this" stuff is that you're going to have more than one person at a time (which person you're dealing with must be clear). That's what "this" is: the instance of the object at hand.

Once you have the template, you can create new instances of the object, like this:

```
myFather=new person("John","Doe",50,"blue");
myMother=new person("Sally","Rally",48,"green");
```

You can also add some methods to the person object. This is also done inside the template:

```
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname;
this.lastname=lastname;
this.age=age;
this.eyecolor=eyecolor;
this.newlastname=newlastname;
}
```

Note that methods are just functions attached to objects. Then we will have to write the newlastname() function:

```
function newlastname(new_lastname)
{
this.lastname=new_lastname;
}
```

The newlastname() function defines the person's new last name and assigns that to the person. JavaScript knows which person you're talking about by using "this.". So, now you can write: myMother.newlastname("Doe").

You Have Learned JavaScript, Now What?

JavaScript Summary

This tutorial has taught you how to add JavaScript to your HTML pages, to make your web site more dynamic and interactive.

You have learned how to create responses to events, validate forms and how to make different scripts run in response to different scenarios.

You have also learned how to create and use objects, and how to use JavaScript's built-in objects.

For more information on JavaScript, please look at our JavaScript examples and our JavaScript reference.
