

**Course material  
on  
Web Technologies  
(Web page –Design/Maintenance)  
JTO2005\_IT specialisation**

## **INDEX**

<b>Content</b>	<b>Page</b>
Markup Language –HTML.....	2
Front Page.....	34
Cascading Style sheets.....	83
XML.....	102
Java Script.....	133
Java.....	178
PHP.....	197
ASP.NET.....	250
Web Services.....	285
Apache.....	290
IIS.....	293
FTP/SFTP.....	301
SSL/SSH.....	306.
Telnet.....	315.
MySQL demo.....	320

# HTML

## What is an HTML File?

- HTML stands for **Hyper Text Markup Language**
- 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**

## Prepare a very simple html page

If you are running Windows, start Notepad. 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 `<b>` and `</b>` 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, Claris Home Page, or Adobe PageMill instead of writing your markup tags in a plain text file.

But if you want to be a skillful Web developer, we strongly recommend that you use a plain text editor to learn your primer HTML. 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 <b> and </b>
- 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**, <b> 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**: <b>  
The **content** of the HTML element is: This text is bold  
The HTML element ends with an **end tag**: </b>

The purpose of the <b> 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>. When you surf the Web, you will notice that most tutorials use uppercase HTML tags in their examples. We always use lowercase tags. Why?

If you want to prepare yourself for the next generations of HTML, you should start using 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.

## Tag Attributes

Tags can have attributes. Attributes can provide additional information about the HTML elements on your page.

This tag defines the body element of your HTML page: <body>. With an added bgcolor attribute, you can tell the browser that the background color of your page should be red, like this: <body bgcolor="red">.

This tag defines an HTML table: <table>. With an added border attribute, you can tell the browser that the table should have no borders: <table border="0">

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

Attributes are always added to the start tag of an HTML element.

## Quote Styles, "red" or 'red'?

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'
```

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.

## 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 `<p>` 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.

## Line Breaks

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

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

The `<br>` tag is an empty tag. It has no closing tag.

## 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.

## 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 `<br>` tag instead. (But don't use the `<br>` tag to create lists. Wait until you have learned about HTML lists.)

You might have noticed that paragraphs can be written without the closing tag `</p>`. Don't rely on it. The next version of HTML will not allow you to skip ANY closing tags.

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.

## Basic HTML Tags

Tag	Description
<code>&lt;html&gt;</code>	Defines an HTML document
<code>&lt;body&gt;</code>	Defines the document's body
<code>&lt;h1&gt; to &lt;h6&gt;</code>	Defines header 1 to header 6
<code>&lt;p&gt;</code>	Defines a paragraph

<code>&lt;br&gt;</code>	Inserts a single line break
<code>&lt;hr&gt;</code>	Defines a horizontal rule
<code>&lt;!--&gt;</code>	Defines a comment

## How to View HTML Source

Have you ever seen a Web page and wondered "Hey! How did do 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
<code>&lt;b&gt;</code>	Defines bold text
<code>&lt;big&gt;</code>	Defines big text
<code>&lt;em&gt;</code>	Defines emphasized text
<code>&lt;i&gt;</code>	Defines italic text
<code>&lt;small&gt;</code>	Defines small text
<code>&lt;strong&gt;</code>	Defines strong text
<code>&lt;sub&gt;</code>	Defines subscripted text
<code>&lt;sup&gt;</code>	Defines superscripted text
<code>&lt;ins&gt;</code>	Defines inserted text
<code>&lt;del&gt;</code>	Defines deleted text
<code>&lt;s&gt;</code>	Deprecated. Use <code>&lt;del&gt;</code> instead
<code>&lt;strike&gt;</code>	Deprecated. Use <code>&lt;del&gt;</code> instead
<code>&lt;u&gt;</code>	Deprecated. Use styles instead

## "Computer Output" Tags

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

## Citations, Quotations, and Definition Tags

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

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: &lt; or &#60;

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.

## 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 &nbsp; character entity.

## The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	quotation mark	&quot;	&#34;
'	apostrophe	&apos;	(does not work in IE)

## Some Other Commonly Used Character Entities:

Result	Description	Entity Name	Entity Number
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
§	section	&sect;	&#167;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
×	multiplication	&times;	&#215;
÷	division	&divide;	&#247;

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

## 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.

## **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.

### **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>
```

## 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.

## Frame Tags

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

## HTML Tables

### 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.

```


| Heading       | Another Heading |
|---------------|-----------------|
| row 1, cell 1 | row 1, cell 2   |
| row 2, cell 1 | row 2, cell 2   |


```

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.

```


|               |               |
|---------------|---------------|
| row 1, cell 1 | row 1, cell 2 |
| row 2, cell 1 |               |


```

**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 (&nbsp;) 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>&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.

### 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.

### 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:**

Coffee

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.

## List Tags

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

## HTML Forms and Input

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

### 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>
<input type="checkbox" name="bike">
I have a bike
<br>
<input type="checkbox" name="car">
I have a car
</form>
```

How it looks in a browser:

I have a bike

I have a car

---

## 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:

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.

### 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.

### 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:

```

```

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:

```

```

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.

## 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.

### 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).

## 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

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 light source is 0 (hex #00). The highest value is 255 (hex #FF).

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

Color	Color HEX	Color RGB
Black	#000000	rgb(0,0,0)
Red	#FF0000	rgb(255,0,0)
Green	#00FF00	rgb(0,255,0)
Deep Blue	#0000FF	rgb(0,0,255)
Yellow	#FFFF00	rgb(255,255,0)
Sky Blue	#00FFFF	rgb(0,255,255)
Pink	#FF00FF	rgb(255,0,255)
Grey	#COCOCO	rgb(192,192,192)
White	#FFFFFF	rgb(255,255,255)

## **Color Names**

A collection of 147 color names are supported by popular browsers.

## **Web Safe Colors**

A few years ago, when most computers supported only 256 different colors, a list of 216 Web Safe Colors was suggested as a Web standard.

## **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.

#### **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.

# HTML 4.01 Quick List

## 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/"></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>
```

---

# 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.

## 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
<code>size="number"</code>	<code>size="2"</code>	Defines the font size
<code>size="+number"</code>	<code>size="+1"</code>	Increases the font size
<code>size="-number"</code>	<code>size="-1"</code>	Decreases the font size
<code>face="face-name"</code>	<code>face="Times"</code>	Defines the font-name
<code>color="color-value"</code>	<code>color="#eefff00"</code>	Defines the font color
<code>color="color-name"</code>	<code>color="red"</code>	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.

## Why use HTML 4.0?

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:

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

```
<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.

## 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 </p>.

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

## 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.

## 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>
```

## Style Tags

Tag	Description
<code>&lt;style&gt;</code>	Defines a style definition
<code>&lt;link&gt;</code>	Defines a resource reference
<code>&lt;div&gt;</code>	Defines a section in a document
<code>&lt;span&gt;</code>	Defines a section in a document
<code>&lt;font&gt;</code>	Deprecated. Use styles instead
<code>&lt;basefont&gt;</code>	Deprecated. Use styles instead
<code>&lt;center&gt;</code>	Deprecated. Use styles instead

# HTML Head

## 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 `<p>` 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
<code>&lt;head&gt;</code>	Defines information about the document
<code>&lt;title&gt;</code>	Defines the document title
<code>&lt;base&gt;</code>	Defines a base URL for all the links on a page
<code>&lt;link&gt;</code>	Defines a resource reference
<code>&lt;meta&gt;</code>	Defines meta information

Tag	Description
<code>&lt;!DOCTYPE&gt;</code>	Defines the document type. This tag goes before the <code>&lt;html&gt;</code> start tag.

## HTML Meta

### 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.

### 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.

### 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  tag points to a place (an address) on the Web with an href attribute value like this: [Last Page](lastpage.htm).

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](http://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:

```
<a href="news:alt.html">HTML Newsgroup</a>
```

## **Downloading with FTP**

The following HTML code:

```
<a href="ftp://www.w3schools.com/ftp/winzip.exe">Download WinZip</a>
```

## **Link to your Mail system**

The following HTML code:

```
<a href="mailto:someone@w3schools.com">someone@w3schools.com</a>
```

creates a link to your own mail system like this:

someone@yahoo.com

## **HTML Scripts**

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

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!

## **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	<i>class_rule or style_rule</i>	The class of the element
id	<i>id_name</i>	A unique id for the element
style	<i>style_definition</i>	An inline style definition
title	<i>tooltip_text</i>	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	<i>language_code</i>	Sets the language code

## Keyboard Attributes

Attribute	Value	Description
accesskey	<i>character</i>	Sets a keyboard shortcut to access an element
tabindex	<i>number</i>	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.

### Window Events

Only valid in body and frameset elements.

Attribute	Value	Description
onload	<i>script</i>	Script to be run when a document loads
onunload	<i>script</i>	Script to be run when a document unloads

### Form Element Events

Only valid in form elements.

Attribute	Value	Description
onchange	<i>script</i>	Script to be run when the element changes
onsubmit	<i>script</i>	Script to be run when the form is submitted
onreset	<i>script</i>	Script to be run when the form is reset
onselect	<i>script</i>	Script to be run when the element is selected
onblur	<i>script</i>	Script to be run when the element loses focus
onfocus	<i>script</i>	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.

This  
is  
a  
text

URL Encode

Bottom of Form

## 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.

## **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.

# **Microsoft FrontPage 2000**

## **Introduction**

Microsoft FrontPage 2000 is a HTML editor available in the MS Office suit .This tutorial will show you how to create attractive Web sites.

This tutorial is divided into two lessons:

### **Lesson 1: Creating and Editing Web Pages**

This lesson teaches you how to create and edit Web pages; work with text and hyperlinks; add pictures, animations, clip art, and files; format lists; position objects; design a feedback form; make a photo gallery; design a web structure; and create a web.

### **Lesson 2: Designing and Publishing a Web**

In this lesson, you will learn how to create navigation hyperlinks; add shared borders and navigation bars to pages; insert page banners; apply and customize a graphical theme; check spelling and replace text across the web; sort and organize files and folders; view web reports; and preview and publish the finished web.

This tutorial assumes that your computer meets the minimum system requirements for FrontPage 2000 .The screen examples in the FrontPage Tutorial assume that you have a standard monitor set to a screen resolution of 800x600, at the minimum. If you are using a different resolution or monitor, the pages you create during the lessons may look slightly different on your screen.  
If you have Web server software installed

The Web site you will create while taking the FrontPage Tutorial will be saved to a folder on your local hard drive. If you upgraded from FrontPage 98 and have the Microsoft Personal Web Server installed, or if you are running Microsoft Internet Information Services (IIS), you can either accept the default destination for the tutorial web, or choose to save it directly on your Web server.

## **FrontPage and Microsoft Internet Explorer**

To get the most out of FrontPage, installing Microsoft Internet Explorer is recommended. When Internet Explorer is installed, FrontPage provides enhanced page and themes preview. Other Web browsers currently do not support these additional features.

## **Learning More About FrontPage**

Throughout the FrontPage Tutorial, you may notice the Help button, as shown here. This button identifies key features and their associated keywords for the online Help system and the Answer Wizard in FrontPage.

FrontPage includes a comprehensive Help system that contains conceptual overviews of key features, step-by-step procedures, and complete, context-sensitive Help. It also contains the Answer Wizard, which you can use to ask questions about features in your own words, such as "How do I check the spelling on my page?"

While taking the FrontPage Tutorial, you may come across terms that are new to you. These terms might appear in the FrontPage user interface, or they may be used in the computer industry and the Internet community. To look up the definitions of words you are unfamiliar with, turn to the Glossary in the FrontPage 2000 Getting Started book.

## **Lesson 1**

### **Creating and Editing Web Pages**

The World Wide Web is a great way for people to communicate with each other. While you have conversations with other people over Internet e-mail and in Internet newsgroups, you can also publish your own personal Web site — a collection of one or more pages on which you can share all sorts of information.

This tutorial will introduce you to Web page creation and Web site management, two aspects that will show you how easy and fun it is to build and maintain a Web site with Microsoft FrontPage 2000.

In this first lesson, you'll create a "Sample Web Page". When you have completed the lessons, you will have a good understanding of FrontPage and its features.

#### **In this lesson, you will learn how to:**

- Start Microsoft FrontPage.
- Create and edit Web pages.
- Work with text and hyperlinks.
- Insert pictures and files.
- Format lists.
- Position objects.
- Add a feedback form.
- Design a photo gallery.
- Create a web structure.
- Save your work.

#### **Starting FrontPage**

If you haven't already installed FrontPage, you'll need to do so before you begin the tutorial.

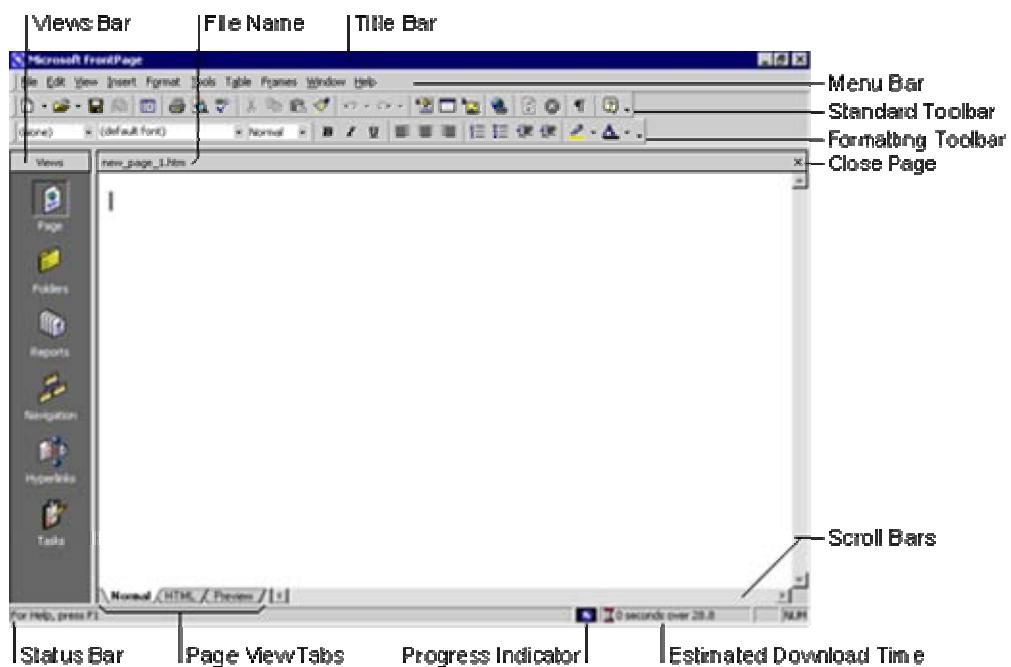
##### **► To start Microsoft FrontPage**

- On the Windows taskbar, click the **Start** button, point to **Programs**, and then click **Microsoft FrontPage**.

FrontPage opens and displays a blank page ready for editing.

#### **Workspace Overview**

FrontPage 2000 has a new, integrated interface that lets you create and edit Web pages and manage entire Web sites — all within one application. All toolbars and menus are consistent with Microsoft Office applications and can be fully customized. You can also use convenient keyboard shortcuts to accelerate common tasks such as opening webs and pages, printing, and many other commands.



The Standard and Formatting toolbars are displayed by default. They provide easy access to the commands you will use most often when working in FrontPage.

What you see in the main application window depends on the currently selected view.

The icons on the Views bar provide different ways of looking at the information on your page or in your web. As you work with FrontPage, you'll frequently switch between views to match the current task at hand — from the early steps of creating a page, to the moment a whole Web site is ready to be published on the World Wide Web.

When you start FrontPage, Page view is displayed by default. Page view is a powerful editing tool for creating and designing Web pages. As you enter text, place pictures, insert documents, create tables, make lists, and design the appearance of your Web pages, Page view displays them as they will appear in a Web browser. All HTML code is automatically created in the background and you don't need to manually edit any code unless you want to.

You'll begin working in Page view and learn about the other views later in this lesson.

## Getting Started

For this tutorial, you'll create a Web site with five pages, on which you will tell visitors about the new millennium. Until it is published for the first time, a Web site is a work in progress. If the task of putting together a whole site seems daunting to you, don't worry. You can gradually add information and other pages to your web until it is finished. Unlike printed letters, memos, and word-processing documents, Web sites can be dynamically changed or updated even after they've been published. You can add, delete, and modify text, pictures, and entire pages at any time.

With FrontPage, you can get started easily by simply typing text on the blank document that Page view provides. For this lesson, we'll begin with the home page — the default document that greets your visitors when they surf to your Web site.

### ► To create a home page

The home page is the front door to your Web site. Greeting your visitors as you might do in person and providing some information about the content or subject matter of your site will spark

interest in the people looking at your site. The home page also contains links to the other pages in a web.

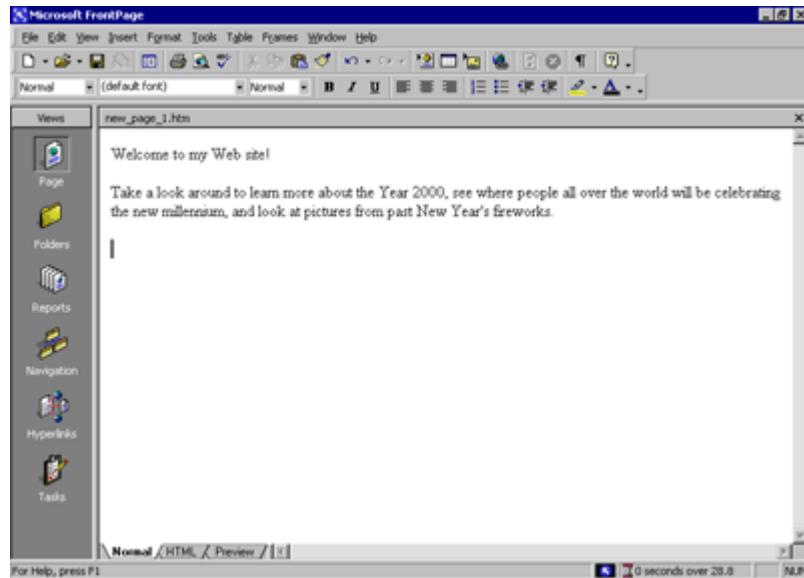
1. On the blank page in Page view, type **Welcome to my Web site!** and then press ENTER.

Just like in a word processor, pressing ENTER puts the cursor on a new line.

2. Next, type the sentence **Take a look around to learn more about the Year 2000, see where people all over the world will be celebrating the new millennium, and look at pictures from past New Year's fireworks.**
3. Press ENTER.

After typing such a long sentence, you may wonder how much typing you'll need to do before getting to the fun stuff. Don't worry, for most of the Millennium Celebration Web content, we've already done the typing for you. And when you're ready to make your own Web site, FrontPage lets you import any of your existing documents directly onto your Web pages without having to retype anything.

Your page should now look like this:



Next, you will add a picture to the bottom of the current page. Pictures can be scanned photographs, drawings, or computer graphics created in a drawing or image-editing program.

For this example, the picture you'll insert is a graphical button of the FrontPage 2000 web logo.

#### ► **To insert a picture on the home page**

1. On the **Insert** menu, point to **Picture**, and then click **From File**.
2. FrontPage displays the **Picture** dialog box. Because you are editing a page that isn't part of a web yet, FrontPage also opens the **Select File** dialog box, which lets you choose a picture to insert from your local file system.

The picture file you'll insert is located in the Tutorial folder that was installed with the FrontPage program files.

3. In the **Select File** dialog box, navigate to the folder named Program Files\Microsoft Office\Office\Tutorial by double-clicking each folder in this path until the **Look in** box displays the Tutorial folder.

If you downloaded the tutorial files from the www.microsoft.com, navigate to the folder named Fptutor\Samples, or to the folder where you placed the files.

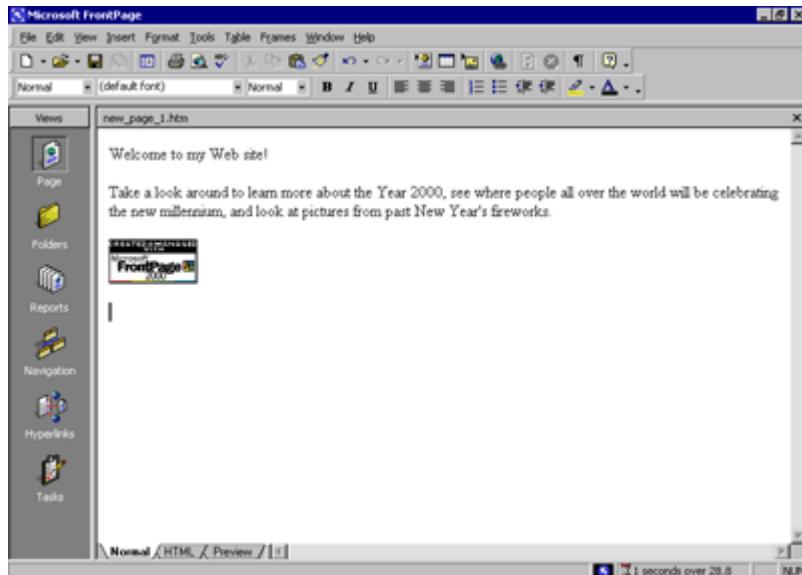
In the Tutorial folder, several files will be displayed. By default, FrontPage searches for picture files.

4. Click the file named **fp2000**, and then click **OK**.

FrontPage inserts the selected picture file on the current page. It is a picture of a button that your site visitors will be able to click to learn more about FrontPage 2000.

5. Press ENTER to create a new line.

Your page should now look like this:



Merely inserting a picture of a button doesn't mean that anything will happen when someone clicks it in a Web browser. To make a picture or a word "clickable," it must have a hyperlink associated with it.

A hyperlink is a pointer from text or from a picture to another page or file on the World Wide Web. On the World Wide Web, hyperlinks are the primary way to navigate between pages and Web sites.

In the next steps, you'll create a hyperlink from the button you just placed on the home page.

► **To create a hyperlink from a picture**

1. On the home page, click the picture of the FrontPage 2000 button you previously inserted.

When a picture is selected, it is shown with file handles — eight small squares around the outline of the picture. These can be used to resize a picture or change its appearance. When a picture is selected, FrontPage also displays the **Pictures** toolbar below Page view. The **Pictures** toolbar provides picture editing and formatting tools, which you'll learn about later.

2. On the **Insert** menu, click **Hyperlink**.

FrontPage displays the **Create Hyperlink** dialog box. Here, you specify the target of the hyperlink you are creating. This can be a page or a file in your web, on your local file system, on a Web server, or on another site on the World Wide Web.

Because you're creating a hyperlink from a button that is labeled "Microsoft FrontPage," you'll link to the Microsoft FrontPage home page on the World Wide Web. When site visitors click the button in a Web browser, they will be taken to the right place.

3. In the **URL** box, type **www.microsoft.com/frontpage** immediately after the **http://** prefix that FrontPage has provided for you.

URL is an acronym for Uniform Resource Locator. It is the technical term for what's commonly known as an "Internet address" or "Web site address." A URL specifies the unique location of a file or a collection of files on the World Wide Web.

4. Click **OK** to finish creating the hyperlink.

You may notice that the appearance of the button itself hasn't changed. Unlike text hyperlinks, which change the color of the clickable text and underline it, picture hyperlinks do not automatically indicate the presence of the hyperlink in an obvious way. This is intentional, because changing the appearance of the picture could obscure the intended page design in some cases.

You can quickly check the existence of a hyperlink from a picture by moving the mouse pointer over the picture. If a hyperlink is present, FrontPage displays the URL the hyperlink points to on the status bar.

Next, you'll insert an animation of the number 2000 at the top of the page. Animated pictures are inserted in the same way as normal pictures.

► **To insert an animated picture on the home page**

1. Press **CTRL+HOME** to quickly jump to the beginning of the current page.

The key combination **CTRL+HOME** places the cursor in the home position — the top left margin on the current page.

2. On the **Insert** menu, point to **Picture**, and then click **From File**.

This time, FrontPage immediately displays the contents of the Tutorial folder. For the duration of each work session, FrontPage remembers the names and locations of the folders you've already navigated to.

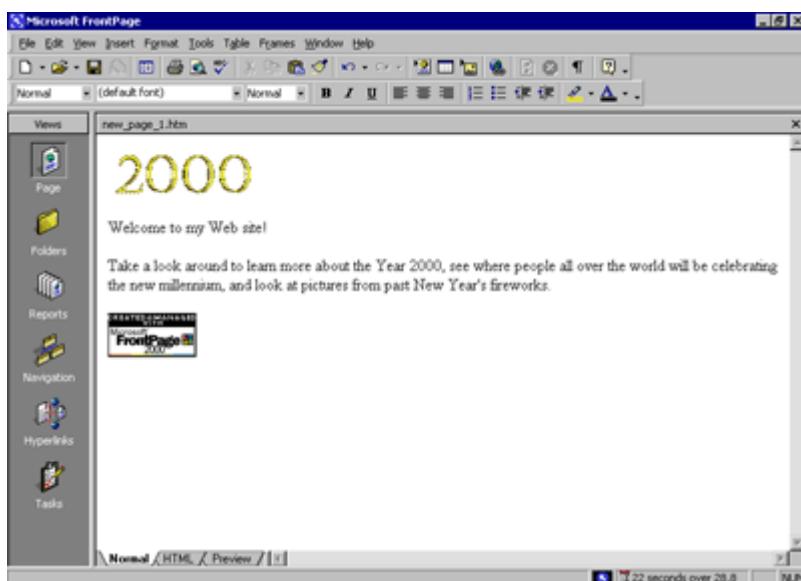
3. In the **Select File** dialog box, double-click the file named **2000**.

Double-clicking file names is faster than selecting each file and clicking **OK**.

FrontPage inserts the animated picture of the number 2000 on the current page. This picture will look great against a dark background, which you'll apply later. While you edit pages in Page view, FrontPage purposely does not show text or picture animations, so they don't distract you from your work. You will be able to see what the animation looks like when you preview the home page later in this lesson.

4. Press ENTER to move the welcome text to the line below.

Your page should now look like this:



To finish the home page, you'll center the text and pictures on it.

► **To center elements on a page**

1. On the **Edit** menu, click **Select All**. FrontPage selects everything on the current page.
2. On the **Format** menu, click **Paragraph**.

FrontPage displays the **Paragraph** dialog box. Here, you can change the alignment of selected elements, and apply indentation and custom spacing for text and graphics.

3. In the **Alignment** list, click **Center**, and then click **OK**. FrontPage centers the text and the pictures on the home page.
4. Click anywhere on the page to deselect all page elements.

Now that you've invested some time and completed a number of steps, it's a good idea to save your page.

► **To save the current page**

1. On the **File** menu, click **Save As**. FrontPage displays the **Save As** dialog box. Here, you can specify the location for the current page, and review or change the page title, the file name, and the file type.
2. In the **Save As** dialog box, click the **My Documents** icon on the vertical Places bar.

The contents of your My Documents folder is displayed. If no files are displayed in the file list, then you currently do not have any other Web pages stored here.

3. Next to the **Page title** field, click the **Change** button.

FrontPage displays the **Set Page Title** dialog box. Here, the default page title is based on the first line of text on the current page. A title identifies the contents of a page when it is displayed in a Web browser. For this tutorial, you'll change the page title to something more descriptive.

4. In the **Set Page title** box, type **Millennium Celebration – Home Page** and then click **OK**.
5. In the **Save As** dialog box, the default file name is based on the first line of text on the current page. For this tutorial, change the file name to something more descriptive.
6. In the **File name** box, change the suggested text to **homepage**, and then click **Save**.

FrontPage saves the current page.

### **Page View Options**

While creating the home page, you've worked exclusively in normal Page view, but there are three different ways you can choose to look at the current page.

► **To display HTML tags on the current page**

- In Page view, click **Reveal Tags** on the **View** menu.

FrontPage displays graphical representations of standard HTML tags for the current page. This display is useful for people who want to know where HTML tags are placed while they design their pages.

- To hide the tags, click **Reveal Tags** on the **View** menu a second time.

### **To display the HTML of the current page**

- In Page view, click the **HTML** tab at the bottom of the page.

This is the HTML code that FrontPage has created so far while you were designing the home page. Web browsers decode these instructions to display the page. The **HTML** tab in Page view is

intended for experienced web developers and page designers who want to customize the HTML that FrontPage creates.

If you want to set your preferences for the way FrontPage will generate HTML code, click Page Options on the Tools menu, and then click the HTML Source tab. If you're not experienced in HTML, you don't need to make any changes here. Click **Cancel** to close the **Page Options** dialog box.

- Click the **Normal** tab at the bottom of the page to return to normal Page view.

#### To preview the current page

- In Page view, click the **Preview** tab at the bottom of the page.

Note: If you do not have Microsoft Internet Explorer installed on your computer, the **Preview** tab will not be displayed, and you will not be able to preview your pages this way.

Looking at your page on the Preview tab is a quick and convenient way to see how certain elements — including animations, movie clips, tables, and lists — will appear in a Web browser.

When you preview the home page you've created, you can see what the animation at the top of the page looks like. Each of the four digits rotates into place one by one, until the number 2000 is displayed. The color and edges of the four digits will look great against a dark background, which you will add in the next lesson. The animation effect will be repeated for as long as the current page is previewed or displayed in a Web browser.

- Click the **Normal** tab at the bottom of the page to return to normal Page view once again.

#### Creating a Web with FrontPage

A web is the collection of a home page and its associated pages, graphics, documents, multimedia, and other files. Webs are stored on a Web server or on a computer's hard drive. FrontPage-based webs also contain files that support FrontPage-specific functionality and allow webs to be opened, copied, edited, published, and administered with FrontPage.

In the previous procedures, you learned how easy it is to create a Web page with FrontPage. As soon as you start the application, you can start typing and editing, then save the document to your hard drive — much like a word processor. While you can certainly choose to put together an entire Web site like this, it can take a lot of manual work and attention to detail to maintain hyperlinks and source files, and keep your content up to date.

When you save your pages to a web, FrontPage can automatically manage and repair hyperlinks, organize files and folders, maintain dynamic navigation bars, check spelling across all pages in the web, and generate reports that point out problems with your pages and files.

#### To create a new web

1. On the **File** menu, click **Close** to close the current page.
2. On the **File** menu, point to **New**, and then click **Web**.

FrontPage displays the **New** dialog box. Here, you can choose from several web templates and wizards, specify where you want to save your web, and what you want to call it.

3. Make sure the **One Page Web** template is selected, and then press TAB.

Pressing the TAB key moves the selection to the field where you specify the name and location of the new web.

4. In the **Specify the location of the new Web** box, change the suggested name to **C:\My Documents\My Webs\Millennium** and then click **OK**.

FrontPage creates a new web named "Millennium" and displays its name in the title bar at the top of the FrontPage application window. Because you'll be working with several files in your web, FrontPage also displays the Folder List, where you can see the files and folders in your current web, similar to files and folders in Windows Explorer. You'll learn how to use the Folder List later, in Lesson 2.

5. Click the **Navigation** icon on the Views bar.

When you have a web open, the icons on the Views bar let you look at the information in your web in different ways.

Navigation view shows a graphical representation of the structure of your Web site. Because you created a one-page web, FrontPage has automatically designated it as the web's home page — indicated with a small icon of a house. While in Navigation view, FrontPage also displays the Navigation toolbar, which you can drag anywhere on your screen.

Next to the Views bar, FrontPage displays the optional Folder List, just like it did in Page view.

In a moment, you'll replace the new, empty home page with the one you created earlier in this lesson. First, however, you'll create the structure for the other four pages that the Millennium Celebration Web will have.

Creating a web structure in Navigation view enables features such as page banners and navigation bars that are automatically updated whenever you change, add, or remove pages in your web. This makes it easy to change things around. You'll learn more about these features later.

#### To create a navigation structure

1. In Navigation view, click the **New Page** button on the toolbar.

FrontPage creates a new page labeled "New Page 1" below the home page. Pages in Navigation view aren't the actual pages in the current web; they are placeholders that point to them. This way, you can easily experiment with the structure and organization of a web before you create its content.

2. To quickly create the remaining three pages, hold down CTRL on your keyboard and press N three times.

CTRL+N is a keyboard shortcut for the New Page command. FrontPage supports common Windows and Microsoft Office accelerator keys that help speed up repetitive tasks. The pages you just created appear below the home page, because the home page was selected when you issued the command.

In Navigation view, the selected page is blue, while others are yellow.

3. With the home page still selected, press TAB.

Pressing the TAB key moves the selection to the next page in the structure and activates the page title for editing.

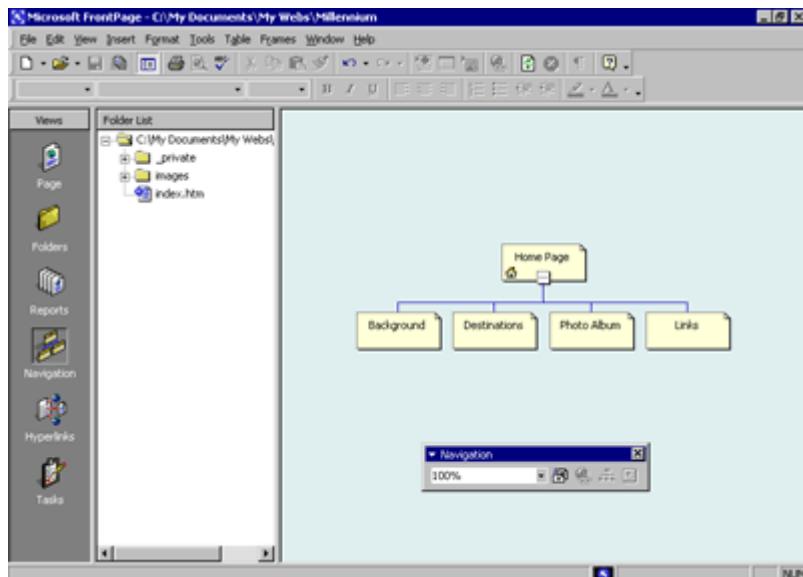
4. When **New Page 1** is selected, type **Background** and then press TAB.

"Background" is the page title of one of the pages you'll create for the Millennium Celebration Web. Next, you'll specify the page titles for the other pages.

5. When **New Page 2** is selected, type **Destinations** and then press TAB.
6. When **New Page 3** is selected, type **Photo Album** and then press TAB.
7. Finally, when **New Page 4** is selected, type **Links** and then press ENTER.

Pressing ENTER after editing a page title saves the new title without selecting another page. To deselect all pages, click anywhere outside the pages in Navigation view.

**Your screen should now look like this:**



You can quickly open pages in Page view for editing by double-clicking the pages in Navigation view or in the Folder List.

Next, you'll replace the blank home page FrontPage created from the web template by importing the home page you created and saved to your My Documents folder earlier in this lesson.

► **To import a page into a web**

1. In Navigation view, double-click the **Home Page** to open it in Page view.

FrontPage opens the blank home page that was created from the web template.

2. On the toolbar, click the **Folder List** button to hide the Folder List in Page view.
3. On the **Insert** menu, click **File**.

FrontPage displays the **Select File** dialog box. Here, you can insert Web pages, word-processing documents, text files, and other documents on the current page.

4. In the **Select File** dialog box, navigate to the My Documents folder.
5. When **My Documents** is displayed in the **Look in** list of the **Select File** dialog box, click the file named **homepage**, and then click **Open**.

**FrontPage imports your previously saved home page to the current page.**

6. To save the current page to your web, click **Save** on the **File** menu, or click the **Save** button on the toolbar.

FrontPage displays the **Save Embedded Files** dialog box. Here, you can preview, rename, save, and update embedded files that the current web will use.

When you previously saved this page to the My Documents folder on your file system, FrontPage left the two pictures you inserted in their original location — the FrontPage Tutorial folder. The home page merely pointed to the picture files without copying them to the same folder the page was saved to. To keep Web sites portable, however, you should always keep associated pages and files as part of the web that uses them.

7. In the **Save Embedded Files** dialog box, click **OK**.

FrontPage saves the home page as Index.htm and saves copies of the embedded picture files, 2000.gif and Fp2000.gif, to the current web.

## **Creating Web Content**

Now that the home page is part of the current web, you will create the content for the other pages in the Millennium Celebration Web.

### ► **To edit the Background page**

1. Click the **Navigation** icon on the Views bar to switch back to Navigation view.

Note that the Folder List now shows the two picture files you saved to the current web. The file Index.htm is the new home page. You can later discard the remaining copy of the home page from your My Documents folder.

In the Folder List, the file names of the other pages were automatically derived from the page titles you typed into the pages in Navigation view. For this tutorial, we won't change the names.

2. Double-click the **Background** page to open it in Page view.

This page will provide some background about the new millennium and the Year 2000 for site visitors. For this tutorial, we have provided this text for you, so you can simply insert it on the page without having to type it.

3. When the blank page is displayed in Page view, click **File** on the **Insert** menu.
4. In the **Select File** dialog box, navigate to the folder named Program Files\Microsoft Office\Office\Tutorial by double-clicking each folder in this path until the **Look in** box displays the Tutorial folder.

If you downloaded the tutorial files from the www.microsoft.com, navigate to the folder named Fptutor\Samples, or to the folder where you placed the files.

5. Next, click the **Files of type** list and click **Text Files (\*.txt)** to display the text files in the Tutorial folder.

FrontPage displays the single file matching the criteria.

6. Click **year2000** in the list, and then click **Open**.

The text you are inserting isn't saved in HTML format, so FrontPage displays the **Convert Text** dialog box to let you control how the text will be imported.

7. In the **Convert Text** dialog box, click **Normal paragraphs with line breaks**, and then click **OK**.

FrontPage imports the text file and places it at the insertion point on the Background page.

8. On the toolbar, click the **Save** button to save changes to the Background page.

Next, you will prepare the page titled Destinations. It will tell site visitors about popular travel destinations that many people will visit to celebrate the new millennium. On this page, you will also provide a feedback form that collects travel ideas from people browsing the Millennium Celebration Web.

#### ► To edit the Destinations page

1. On the toolbar, click the **Folder List** button to show the Folder List in Page view.
2. Double-click **destinations.htm** in the Folder List to open the page in Page view.
3. Click the **Folder List** button to hide the Folder List.
4. When the blank page is displayed in Page view, click **File** on the **Insert** menu.
5. In the **Select File** dialog box, click the **Files of type** list, and then click **Rich Text Format (\*.rtf)** to display the formatted text file in the Tutorial folder.

FrontPage displays the file matching the criteria.

6. Double-click the file **events**.

Because this type of file contains formatting, FrontPage automatically converts the formatted text to HTML format.

7. On the toolbar, click the **Save** button to save changes to the Destinations page.

### **Automatic Spelling Checks**

Take a moment to scroll to the top of the Destinations page. In the first paragraph, note the red, wavy underline under the name "Balleny," the name of an island in Antarctica.

In Page view, FrontPage automatically checks the spelling of text you type on the current page, just like Microsoft Word, PowerPoint, and other Microsoft Office applications do. An underlined word doesn't necessarily mean the word is spelled incorrectly. FrontPage may simply prompt you to verify unknown or suspected words, which happens most commonly with names of people and places.

If you know that the spelling of a suspected word is correct, you can either choose to ignore such words and keep them unchanged, or add them to a custom dictionary that FrontPage will keep for subsequent spelling checks. If the word is indeed misspelled, you can quickly insert the corrected spelling by right-clicking the suspected word and selecting a suggested correction.

There are three ways FrontPage can check spelling for you:

- Automatic spelling check as you type text
- Manual spelling check of the current page
- Cross-web spelling checks of all pages in a web

Page-based spelling checks are available in Page view by right-clicking suspect words or by clicking the Spelling command on the Tools menu. Cross-web spelling checks are available in every web view.

Spelling checks are important if you want your web content to give visitors a professional impression. If words are misspelled on a single page, people might question the accuracy of your entire Web site content. The flexible spelling features in FrontPage give you the option of checking spelling page by page as you create and edit content, or doing it all at once, just before you publish your web to the World Wide Web.

For this tutorial, you'll check the spelling of your entire web later, in Lesson 2.

### **Designing a Page**

The Background page will inherit its formatting from a graphical theme that you will apply to the Millennium Celebration Web later, in Lesson 2. The Destinations page, however, requires some more design work.

To help the reader differentiate the paragraph headings, list of travel destinations, and event details that the text on this page talks about, you will add some pictures, format paragraph styles, and create a bulleted list.

#### **► To create a bulleted list**

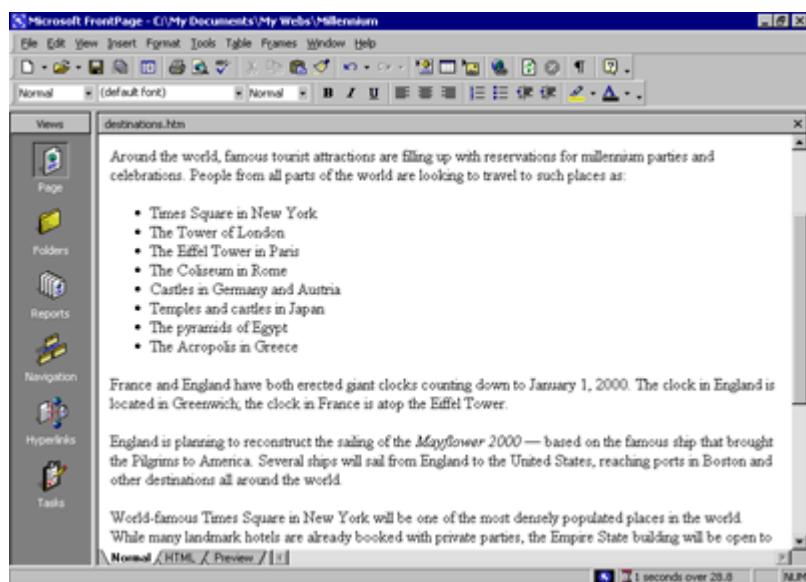
1. With the Destinations page still displayed in Page view, find the words "Times Square in New York."
2. Use the scroll bar to bring the entire list of destinations into view, beginning with "Times Square in New York" and ending with "The Acropolis in Greece."

3. Click just to the left of the letter T in "Times Square," hold down SHIFT, click just after the word "Greece," and then release SHIFT.
4. On the toolbar, click the **Bullets** button.

FrontPage converts the selected text to a bulleted list.

5. Click anywhere on the page to deselect the list.

Your page should now look like this:



You can also create numbered lists with FrontPage. When you add new items to a numbered list, FrontPage automatically numbers them sequentially. You can add to bulleted and numbered lists by pressing ENTER after an item in the list. To end a list, press ENTER twice after typing the last list item.

Next, you will place four pictures on the current page and use positioning features in FrontPage to align the pictures with the paragraphs they are associated with. This will create a more interesting page layout.

#### ► To position pictures with text

1. With the Destinations page still displayed in Page view, scroll down to the words "France and England" in the paragraph just following the list.
2. Click just to the left of the letter F in "France," press HOME, and then click the **Insert Picture From File** button on the toolbar.

When you last inserted a picture, you did not have a web open, and FrontPage automatically displayed the **Select File** dialog box. Now that a web is open, FrontPage assumes you want to work with pictures that are already part of your web, and therefore displays the **Picture** dialog box.

3. Because you haven't yet added the picture you want to the current web, click the **Select File** button in the **Picture** dialog box.

FrontPage displays the **Select File** dialog box.

4. Click the file named **paris**, and then click **OK**.

FrontPage inserts a picture of the Eiffel Tower in Paris just before the current paragraph.

5. Next, click the picture of the Eiffel Tower to select it.

6. On the **Format** menu, click **Position**.

FrontPage displays the **Position** dialog box.

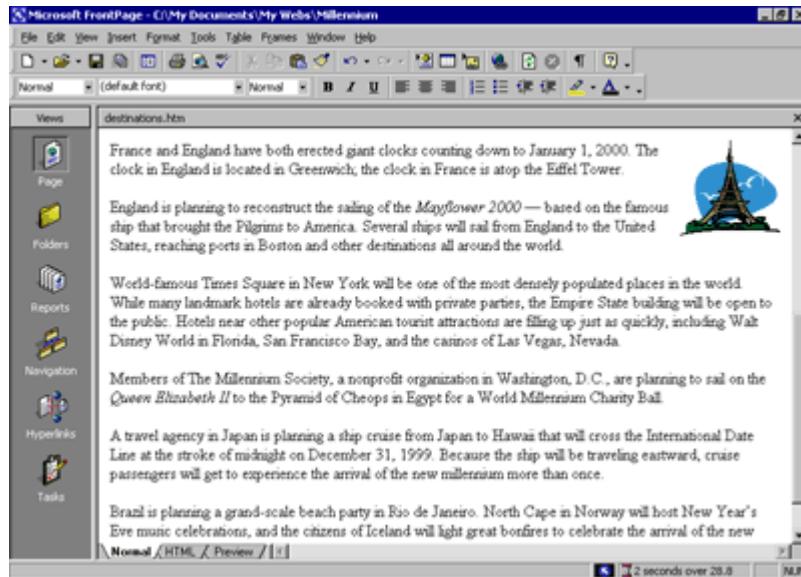
7. Under **Wrapping style**, click **Right**, and then click **OK**.

The picture is aligned with the right margin of the current page and the text now flows around it.

8. On the toolbar, click the **Save** button to save changes to the Destinations page.

9. In the **Save Embedded Files** dialog box, click **OK**.

Your page should now look like this:



You can either place pictures one by one in this way, or you can import all the pictures you will use on your pages all at once. While importing single files is done in Page view, inserting a group of files or entire folders is done in Folders view.

#### ► To add a group of files to the current web

1. Click the **Folders** icon on the Views bar to switch to Folders view.

Folders view is an expanded view of the Folders List that you have seen in Navigation and Page view. Similar to the way you look at files in Windows Explorer, here you can view details about the files and folders in your web, and perform such file management tasks as adding, deleting, moving, copying, and renaming files.

2. On the **File** menu, click **Import**.

FrontPage displays the **Import** dialog box. Here, you can add files and folders from your local file system, a local area network, a company file server, or a resource on the Internet or World Wide Web, such as an FTP server.

3. In the **Import** dialog box, click **Add File**.
4. In the **Add File to Import List** dialog box, navigate to the folder named Program Files\Microsoft Office\Office\Tutorial by double-clicking each folder in this path until the **Look in** box displays the Tutorial folder.

If you downloaded the tutorial files from the www.microsoft.com, navigate to the folder named Fptutor\Samples, or to the folder where you placed the files.

5. Next, click the **Files of type** list, and then click **GIF and JPEG (\*.gif, \*.jpg)** to display all picture files in the Tutorial folder.
6. Click the file named **firewks1** in the list to select it.
7. Next, hold down CTRL, and while doing so, click to select the files named **firewks2, firewks3, firewks4, japan, london, and sanfran**.

FrontPage supports standard Windows file selection methods.

8. When the files are selected in the **Add File to Import List** dialog box, release CTRL, and then click **Open**.

FrontPage adds the pictures you selected to the list in the **Import** dialog box.

9. Click **OK** to import the listed files to the current web.

Now that the remaining pictures are added to your web, it's time to finish the layout of the Destinations page.

#### ► **To finish the page layout**

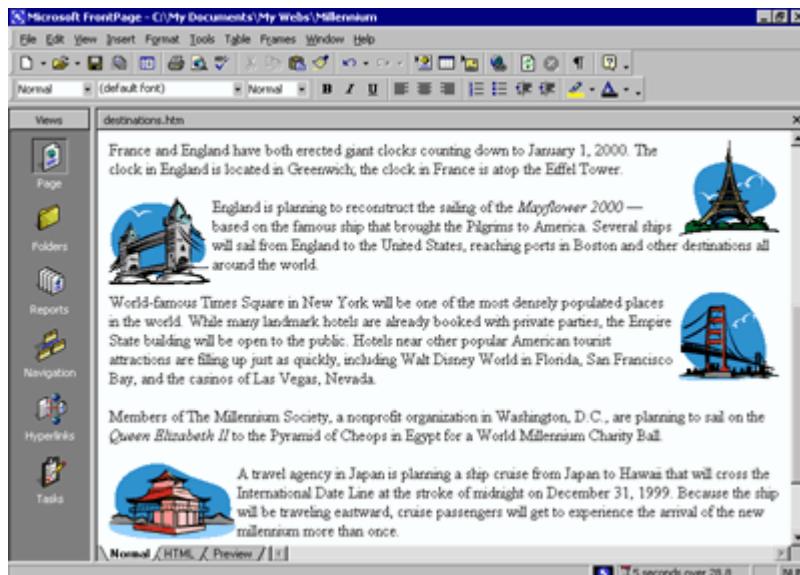
1. Click the **Page** icon on the Views bar to return to Page view.
2. Locate the sentence beginning with "England is planning to reconstruct...," click just to the left of the letter E in "England," and then click the **Insert Picture From File** button on the toolbar.
3. In the **Picture** dialog box, note that your previously imported pictures are now available, and then select the file **london.gif** from the list.

When you click a single picture file in the **Picture** dialog box, FrontPage displays a preview of the picture, so you can make sure it's the one you want to insert. This is one of the benefits of adding all your pictures to the web before inserting them on your pages.

The **Picture** dialog box also links to the clip art gallery that is included with FrontPage. And if you have a scanner or a digital camera, you can click the **Scan** button in this dialog box to acquire original pictures from those sources.

4. Click **OK** to insert the picture
5. Click the picture of the Tower of London you just inserted, and then click the **Align Left** button on the toolbar to position the picture in the left margin and make the text wrap around it.
6. Next, scroll down to the sentence beginning with "World-famous Times Square...," click just to the left of the letter W in "World-famous," and then click the **Insert Picture From File** button on the toolbar.
7. Select the file **sanfran.gif** from the list, and then click **OK**.
8. Click the picture of the Golden Gate Bridge, and then click the **Align Right** button on the toolbar to position the picture in the right margin and make the text wrap around it.
9. Finally, scroll down to the sentence beginning with "A travel agency in Japan...," click just to the left of the letter A at the beginning of the sentence, and then click the **Insert Picture From File** button on the toolbar.
10. Double-click the file **japan.gif** on the list.
11. Click the picture of the Japanese temple, and then click the **Align Left** button on the toolbar to position the image in the left margin and make the text wrap around it.

Your page should now look like this:



Positioning pictures and other page elements around text on your page makes for a more interesting design, much like pages in a magazine or newspaper. By positioning pictures in the margin, your page layout will be preserved even when the page is viewed at a different screen size and resolution in a Web browser.

To finish the Destinations page, you will create a feedback form so that you can interact with site visitors who want to participate. A feedback form can be used to collect comments and information from people visiting your Web site.

#### ► To create a feedback form

1. In Page view, press CTRL+END to quickly jump to the bottom of the current page, or scroll all the way down using the scroll bar.
2. On the new, blank line, type **Tell Us Where You'll Be!** and then press ENTER.
3. On the next line, type **Do you have great travel plans for a millennium party? Then tell us about them below!**
4. Press ENTER to create a new line.
5. On the **Insert** menu, point to **Form**, and then click **Form**.

FrontPage inserts a new form on the current page. The dashed lines indicate the form's boundary. By default, a new form contains Submit and Reset push buttons.

Next, you will customize the default form by adding form-fields and form-field labels, to let site visitors know what kind of information you want them to enter into the form.

#### ► To customize the form

1. With the cursor still positioned to the left of the Submit button, click the **Center** button on the toolbar, and then press ENTER to add some space to the form.
2. Press the UP ARROW key on your keyboard to return the cursor to the beginning of the form.

FrontPage moves the cursor to the middle of the first line of the form.

3. On the first line, type **Your Name:** and then press SHIFT+ENTER.

Holding SHIFT while pressing ENTER creates a line break. Line breaks are useful for spacing lines of text more closely together than standard paragraph spacing.

4. On the **Insert** menu, point to **Form**, click **One-Line Text Box**, and then press ENTER.

FrontPage inserts a one-line text input field into the form.

5. On the next line, type **Your E-mail Address:** and then press SHIFT+ENTER.
6. On the **Insert** menu, point to **Form**, click **One-Line Text Box** once more, and then press ENTER.
7. On the next line, type **Your travel plans are:** and then press SHIFT+ENTER.

8. On the **Insert** menu, point to **Form**, and then click **Scrolling Text Box**.

FrontPage inserts a scrolling text input field into the form.

9. Double-click the scrolling text box you just inserted.

FrontPage displays the **Scrolling Text Box Properties** dialog box. Here, you can change the appearance of the text box.

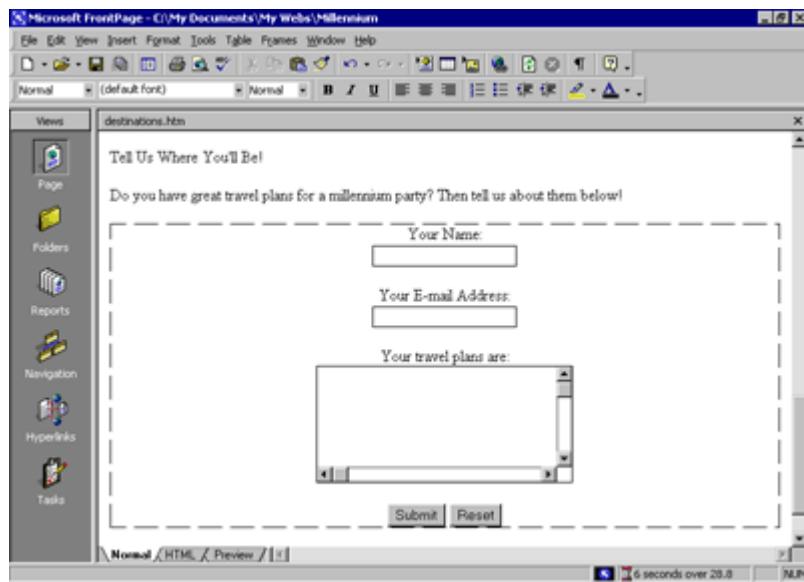
10. In the **Scrolling Text Box Properties** dialog box, change the **Width in characters** to **30** and the **Number of lines** to **5**, and then click **OK**.

The scrolling text box has increased in size, which will encourage site visitors to write a little more than just a brief line of text about their New Year's plans.

Now that your form and the Destinations page are finished, it's a good idea to save your work.

11. On the toolbar, click the **Save** button to save changes to the Destinations page.

Your page should now look like this:



Good work! The feedback form is finished and so is the Destinations page. In the next part of the lesson, we'll add the last two pages — an online photo album and a list of links to your favorite sites on the World Wide Web.

## Creating a Photo Album

The World Wide Web has a graphical interface, so it's no surprise that the most popular Web sites have pictures to look at. Scanners and digital cameras have become much more affordable, and many photo-developing places now offer to put your pictures on a CD-ROM so you can share them online.

While working with the graphics and clip art in the previous procedures, you saw how easy it is to place pictures on Web pages using FrontPage. In this part of the lesson, you'll create an online

photo album of actual photographs. For the Millennium Celebration Web, you'll share some great photos of fireworks.

► **To edit the Photo Album page**

1. On the toolbar, click the **Folder List** button to show the Folder List in Page view.
2. Double-click **photo\_album.htm** in the Folder List to open the page.
3. Click the **Folder List** button to hide the Folder List.
4. When the blank page is displayed in Page view, type **New Year's Fireworks** on the first line, then press ENTER.
5. On the next line, type **Here are some great pictures from past New Year's celebrations. Click each thumbnail to see the full-size picture, then use your Web browser's Back button to return to this page.**
6. Press ENTER twice to create some space.

Next, you'll place four pictures on the current page. They are already part of the web because you imported them in the previous procedure.

7. On the toolbar, click the **Center** button.
8. Click the **Insert Picture From File** button.
9. In the **Picture** dialog box, select **firewks1.jpg**, and then click **OK**.

FrontPage inserts the first of the four fireworks pictures you will place on this page. Next, insert the other three pictures, one after the other. Don't worry about their large size and proper placement just yet.

10. Repeat steps 8 and 9 with the picture files named **firewks2.jpg**, **firewks3.jpg**, and **firewks4.jpg**.

The remaining pictures are placed on the page, one after the other.

11. On the toolbar, click the **Save** button to save changes to the Photo Album page.

Having lots of large pictures on your Web page for people to look at is great, but not everyone has a fast connection to the Internet. Over a dial-up connection with a modem, pages with large pictures and complex designs can take a long time to download before the Web browser displays the page.

Imagine changing the channel on your TV and having to wait several minutes before you see what's playing on that channel. It's no different on the World Wide Web. There are many other "channels" or Web sites to look at. No matter how interesting your site may be, people may quickly lose interest if it takes too long to download.

On the status bar, FrontPage automatically displays the estimated time it will take for the current page to download over the Internet when the page is opened in a Web browser. The default measurement assumes that your site visitors will have a 28.8K modem connection. You can adjust this measurement for other common connection speeds by right-clicking the red hourglass icon and choosing another connection speed from the shortcut menu.

For this tutorial, we'll leave the download speed at 28.8K. On the current page, you can see that the estimated download time for the Photo Album page is 97 seconds. This means that people who will visit the Millennium Celebration Web will have to wait a minute and a half before they can see the four pictures you've just inserted. That's quite a long time to wait for just four pictures.

By creating thumbnails — small preview images of pictures — you can give your visitors the choice of whether they want to spend time downloading the full-size pictures on your page. FrontPage makes creating thumbnails easy with the Auto Thumbnail tool.

#### ► To create thumbnails of pictures

1. With the Photo Album page still displayed in Page view, press CTRL+HOME to quickly jump to the beginning of the current page, or scroll all the way up using the scroll bar.
2. Click the first fireworks picture to select it.

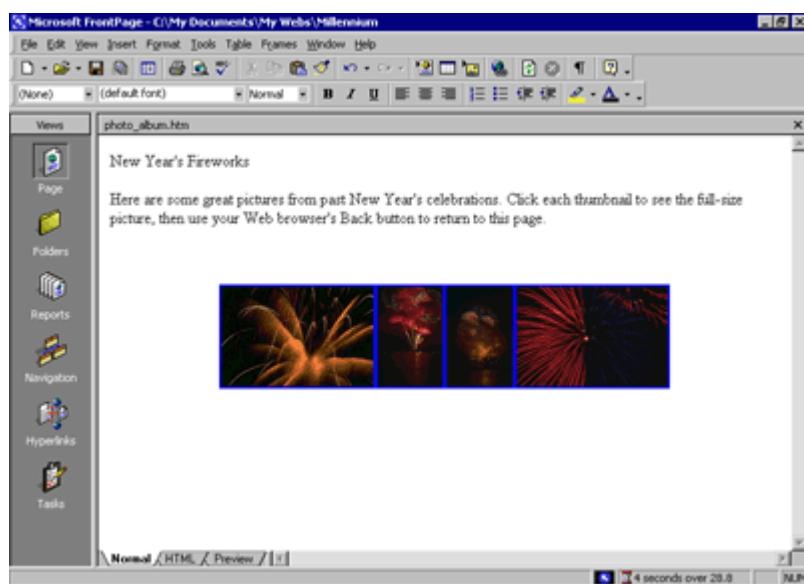
FrontPage displays the **Pictures** toolbar below Page view.

3. On the **Pictures** toolbar, click the **Auto Thumbnail** button.

FrontPage creates a thumbnail of the selected picture and adds a blue border to indicate that it contains a hyperlink to the original picture in your web. When site visitors visit this page, they can click each thumbnail to download the full-size pictures.

4. Repeat steps 2 and 3 for the other three pictures on the Photo Album page.
5. Click the second fireworks thumbnail to select it. Drag one of the top corner handles to make this thumbnail the same height as the first thumbnail. Repeat this step for the third fireworks thumbnail.
6. Click anywhere on the page to deselect the last thumbnail.

Your page should now look like this:



When FrontPage creates thumbnails, it doesn't actually modify the original picture files in any way. Instead, it quickly makes a copy of each picture, resizes it, downsamples the display resolution of the picture, inserts a hyperlink pointing to the original picture file, and adds a border around the thumbnail to indicate the presence of a hyperlink.

You can see why the Auto Thumbnail button is a real timesaver. Doing all of that manually for each picture could take a while.

6. On the toolbar, click the **Save** button to save changes to the Photo Album page.

Because FrontPage made small copies of the pictures that are represented by a thumbnail, it needs to save the thumbnails to the current web. The names of the thumbnail picture files are the same as the original pictures, but FrontPage adds a "\_small" suffix to each file name for easy identification.

7. In the **Save Embedded Files** dialog box, click **OK**.

FrontPage saves the four thumbnails to the Millennium Celebration Web.

Because you have an even number of pictures, you can arrange them a little better than all on one line. You can treat thumbnails like other pictures on your pages and move them where you want them.

► **To finish the Photo Album page**

1. Click the first thumbnail and then press HOME to move the cursor to the left of it.
2. Press the RIGHT ARROW key on your keyboard to move the cursor between the first and second thumbnail.
3. Press the TAB key.

FrontPage creates space between the first and second thumbnail.

4. Next, press the RIGHT ARROW key again to move the cursor between the second and third thumbnail.
5. Press ENTER.

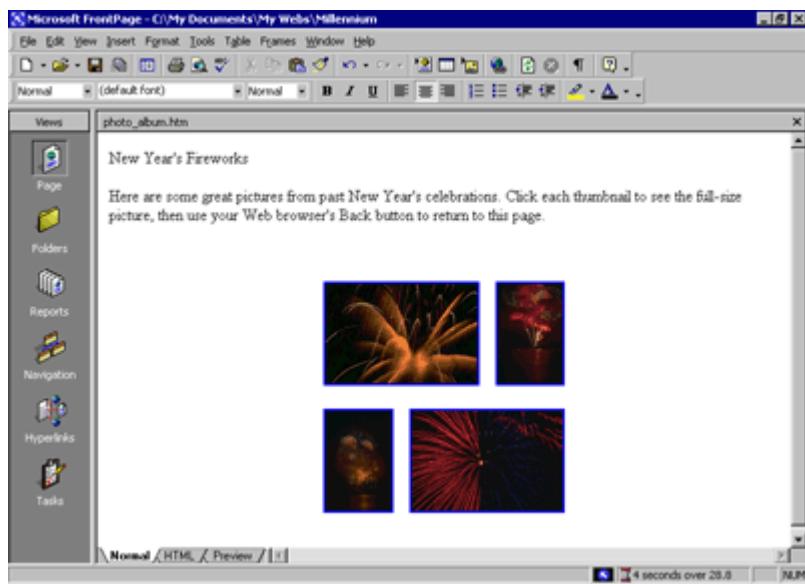
FrontPage moves the third and fourth thumbnail to the next line.

6. Press the RIGHT ARROW key once more to move the cursor between the third and fourth thumbnail.
7. Press the TAB key.

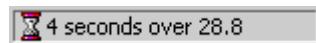
FrontPage creates space between the third and fourth thumbnail.

8. On the toolbar, click the **Save** button to save changes to the Photo Album page.

Your page should now look like this:



Now take another look at the FrontPage status bar. The Photo Album page previously would have taken 97 seconds to download.



After replacing the large pictures with thumbnails, the Photo Album page now only takes 4 seconds to download. That's much better!

## Linking to Other Web Sites

Now only the Links page remains to be edited. For this tutorial, this page will contain a list of text hyperlinks to some other sites on the World Wide Web.

When you create your own webs, you can create hyperlinks pointing to other Web sites that relate to the subject matter of your own pages. This lets visitors browse to similar sites without having to search for them.

### ► To begin the Links page

1. On the toolbar, click the **Folder List** button to show the Folder List in Page view.
2. Double-click **links.htm** in the Folder List to open the page in Page view.
3. Click the **Folder List** button to hide the Folder List again.
4. When the blank page is displayed in Page view, type **Links to My Favorite Sites** and then press ENTER.

Next, you will create a simple text animation of the paragraph heading. FrontPage includes a collection of fun text effects that you can easily apply to text headings or entire paragraphs.

### ► To create a dynamic text effect

1. On the Links page, click anywhere in the text **Links to My Favorite Sites**.
2. On the **Format** menu, click **Dynamic HTML Effects**.

FrontPage displays the **DHTML Effects** toolbar. Here, you make sequential selections that will create a simple dynamic HTML (DHTML) script to animate the text when it is displayed in a Web browser.

Dynamic HTML is an extension of the HTML language that lets you create presentation effects for text and objects, much like in a Microsoft PowerPoint slide show. Using the **DHTML Effects** toolbar, you can add simple effects to your pages without the need to know programming.

1. In the **On** list, click **Page load**.

This will instruct the Web browser to begin the effect when the page loads.

2. In the **Apply** list, click **Hop**.

FrontPage applies the Hop effect. In a Web browser, this effect will cause each word to bounce onto the page.

3. Click the **Close** box in the upper right corner of the **DHTML Effects** toolbar.

The **DHTML Effects** toolbar closes and the dynamic text effect is indicated in Page view with light blue shading.

Next, you will add text hyperlinks pointing to other sites on the World Wide Web. With FrontPage, you can create text hyperlinks in a number of ways, which you will learn next. When you create your own webs, you can create hyperlinks using your preferred method.

#### ► To create hyperlinks from text

1. On the Links page, press the DOWN ARROW key, and then type **MSN - The Microsoft Network** and then press ENTER.
2. Click and drag the mouse over the words you just typed to select them.
3. On the **Insert** menu, click **Hyperlink**.

FrontPage displays the **Create Hyperlink** dialog box. Here, you specify the target of the hyperlink you are creating. This can be a page or a file in your web, on your local file system, on a Web server, or on another site on the World Wide Web.

4. In the **URL** box, type **www.msn.com** immediately after the **http://** prefix that FrontPage has provided for you, and then click **OK**.

"HTTP" stands for Hypertext Transfer Protocol. This is the Internet protocol that allows World Wide Web browsers to retrieve information from Web servers. The text "www.msn.com" is the URL of MSN, the Microsoft Network.

5. On your keyboard, press the DOWN ARROW key to deselect the text.

The words "MSN - The Microsoft Network" have changed from black default text to blue text, and the words are now underlined to indicate the presence of a hyperlink. When this page is displayed in a Web browser, clicking this hyperlink will retrieve and display the MSN home page.

Before creating the next hyperlink, you'll insert a special character symbol to indicate a trademark on the current page.

► **To insert special characters or symbols**

1. Click the mouse just after the letters **MSN** in the hyperlink you just created.
2. On the **Insert** menu, click **Symbol**.

FrontPage displays the **Symbol** dialog box. Here, you can select and insert special characters at the insertion point. You can insert multiple symbols while this dialog box is displayed.

3. In the **Symbol** dialog box, select the trademark (TM) symbol, click **Insert**, and then click **Close**.

FrontPage inserts the trademark symbol after the letters MSN. You can use the **Symbol** command to insert characters that you may not be able to type directly with your keyboard.

Next, you will create an automatic hyperlink. This method of creating hyperlinks is quick and easy, because it lets you bypass the **Create Hyperlink** dialog box.

► **To create an automatic hyperlink**

1. On the Links page, press the DOWN ARROW key, type **http://www.yahoo.com**, and then press ENTER.

Yahoo! is a popular Internet service that lets you look for information on the World Wide Web using search keywords and subject categories.

As soon as you press ENTER, the URL you typed changes from black to blue text and is underlined to indicate the presence of a hyperlink. Like other Microsoft Office applications, FrontPage supports automatic hyperlink creation.

Since a URL by itself is not always very descriptive, however, you'll want to change it to the name of the site that the hyperlink points to. You can overtype the text without erasing the hyperlink.

2. Using the mouse, click and drag over the URL you just typed to select it.
3. When the URL **http://www.yahoo.com** is selected, type **Yahoo!** to replace the selected text.

The hyperlink still points to the same URL, but it is now labeled with the site's name.

Next, you'll create a hyperlink using your Web browser. This method of creating hyperlinks is the most accurate, because you actually visit the page the hyperlink will point to before creating the hyperlink. In addition, FrontPage copies the URL from the Web browser address field, so once the address is verified, you don't have to type it again.

If you do not have access to the World Wide Web while taking the FrontPage Tutorial, skip the following procedure and practice these steps the next time you're connected to the Internet.

► **To create a verified hyperlink**

1. Press the DOWN ARROW key to move the insertion point to the blank line below the previous hyperlink.
2. Type **Microsoft FrontPage 2000** and then press ENTER.
3. Click and drag the mouse over the words you just typed to select them.
4. On the toolbar, click the **Hyperlink** button.

FrontPage displays the **Create Hyperlink** dialog box.

5. In the **Create Hyperlink** dialog box, click the **Web Browser** button.

FrontPage starts your Web browser. When you visit the page that the hyperlink should point to and then switch back to FrontPage, the URL box will contain the address of the target page.

6. In your Web browser's **Address (or Location)** box, type **http://www.microsoft.com/frontpage** and then press ENTER.

The Web browser displays the Microsoft FrontPage home page, where you can learn more about FrontPage, download updates, and find answers to common questions.

7. On your keyboard, press ALT+TAB (or click the Microsoft FrontPage taskbar button on the Windows taskbar) to switch back to the **Create Hyperlink** dialog box.

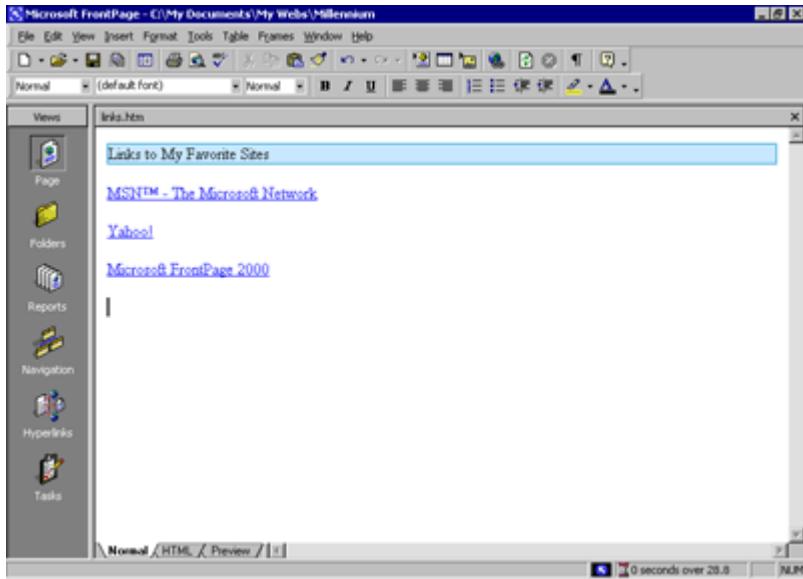
The URL of the Microsoft FrontPage home page is now entered into the **URL** box in the **Create Hyperlink** dialog box.

8. Click **OK**.
9. On your keyboard, press the DOWN ARROW key to deselect the text.

The words "Microsoft FrontPage 2000" are now underlined to indicate the presence of a hyperlink.

10. On the toolbar, click the **Save** button to save changes to the Links page.

Your page should now look like this:



## Formatting Paragraph Headings

Now that you have successfully created the content on all the pages in your web, you'll apply paragraph and font formatting to make the paragraph headings on all pages look consistent.

### ► To apply paragraph styles to headings

1. On the **Window** menu, click **index.htm**.

FrontPage brings the home page back into view. When you have more than one page open in Page view, you can use the Window menu to switch between them. The current page will always be listed at the top of this menu.

2. Click anywhere in the heading **Welcome to my Web site!** at the top of the page.
3. Click the **Style** list on the Formatting toolbar, and change **Normal** to **Heading**

FrontPage applies the Heading 3 style to the current line of text. The size of the text isn't affected, but the text is now bold.

Heading styles in the **Style** list are based on universal HTML standards. A level 1 heading is the largest possible text style for Web pages, and a level 6 heading is the smallest.

4. On the toolbar, click the **Save** button to save the home page.
5. On the **Window** menu, click **background.htm**.

FrontPage brings the Background page back into view.

6. Click anywhere in the heading **The New Millennium** at the top of the page.
7. Click the **Style** list on the toolbar, and then click **Heading 4**.

8. Repeat steps 6 and 7 with the heading **What's in a Number?** on the lower half of the Background page.
9. On the toolbar, click the **Save** button to save changes to the Background page.

► **To repeat paragraph formatting with the Format Painter**

FrontPage provides a convenient way to copy the formatting styles from one selection of text to another with the push of a button. You'll practice how to do this on the Destinations page.

1. On the **Window** menu, click **destinations.htm**.

FrontPage brings the Destinations page back into view.

2. Click anywhere in the sentence **Planning a Millennium Getaway** at the top of the page.
3. Click the **Style** list on the toolbar, and then click **Heading 4**.
4. When the style has been applied, click anywhere in the heading, and then click the **Format Painter** button on the toolbar.
5. Using the scroll bar, scroll down to the heading **Tell Us Where You'll Be**.
6. Click the mouse to drag the **Format Painter** pointer over the words **Tell Us Where You'll Be**, and then release the mouse button.

FrontPage copies the formatting from the previous heading and applies it to the current heading. The Format Painter tool is particularly useful when you want to easily duplicate several concurrent formatting choices from one selection of text to another.

7. On the toolbar, click the **Save** button to save the Destinations page.

**Next, you'll finish formatting the page headings on the remaining two pages.**

► **To finish applying paragraph styles to headings**

1. On the **Window** menu, click **photo\_album.htm**.

FrontPage brings the Photo Album page back into view.

2. Click anywhere in the sentence **New Year's Fireworks** at the top of the page.
3. Click the **Style** list on the toolbar, and then click **Heading 4**.
4. On the toolbar, click the **Save** button to save the Photo Album page.
5. On the **Window** menu, click **links.htm**.
6. Click anywhere in the sentence **Links to My Favorite Sites**.
7. Click the **Style** list on the toolbar, and then click **Heading 4**.
8. On the toolbar, click the **Save** button to save the Links page.

Congratulations, you've successfully completed Lesson 1.

## What's Ahead

In Lesson 2, you'll enhance the appearance of the Millennium Celebration Web by adding shared borders, navigation bars, and a graphical theme, and you will learn how to preview, test, organize, and publish the finished web.

## Finishing Lesson 1

You can continue with Lesson 2 now, or close Microsoft FrontPage and continue the tutorial at a later time.

### ► To close Microsoft FrontPage

- On the **File** menu, click **Exit**.

The FrontPage application closes.

## Lesson 2

### Designing and Publishing a Web

In Lesson 1, you learned how easy it is to create Web pages with Microsoft FrontPage and then add them to a new web.

In this second lesson, you'll continue working with the Millennium Celebration Web you created by adding navigation bars to its pages, applying and customizing a graphical theme, previewing and testing the web, and then preparing the web for publication on the World Wide Web.

Before you publish a web, you'll want to make sure its pages and files are well organized, all of its hyperlinks are working, pages are free of spelling errors, and you have enough space available on the target Web server. FrontPage can help you complete these important tasks.

### In this lesson, you will learn how to:

- Create hyperlinks to other pages within a web.
- Add shared borders and navigation bars.
- Apply and customize a graphical theme.
- Preview and test a web in a Web browser.
- Organize files and folders.
- Generate a Site Summary report.
- Check spelling across all pages in a web.
- Replace text on Web pages.
- Create and assign web tasks.
- Publish a web to the World Wide Web.

## Enhancing the Appearance of a Web

If you're continuing this lesson directly from Lesson 1, the Millennium Celebration Web should still be open in FrontPage. If this is the case, skip down to the procedure named "To create hyperlinks to other pages."

If you're continuing this tutorial from a previous session, then you must first open the web before you can work with its pages.

### ► To open an existing web

1. On the Windows taskbar, click the **Start** button, point to **Programs**, and then click **Microsoft FrontPage**.
2. On the **File** menu, point to **Recent Webs**, and then click **C:\My Documents\My Webs\Millennium** to open the Millennium Celebration Web you created in Lesson 1.

FrontPage opens the web. The application title bar now reads "Microsoft FrontPage – C:\My Documents\My Webs\Millennium."



Because you'll be working with the pages you've already created, you can close the blank page that opened by default in Page view.

3. On the **File** menu, click **Close**, or click the **Close** button in the upper right corner of the page.

FrontPage closes the current page. Page view is now blank, but the Millennium Celebration Web remains open.

While creating hyperlinks from pictures and text in Lesson 1, you may have noticed that you don't have any connections yet between the pages in your web. Even if someone surfed to your current home page, they would have no way of getting to the other pages. In the next section, you'll learn how easy it is to make navigation hyperlinks to other pages.

### ► To create hyperlinks to other pages

1. On the toolbar, click the **Folder List** button to show the Folder List in Page view.
2. Double-click **index.htm** in the Folder List to open the home page in Page view.

You'll keep the Folder List visible while you create hyperlinks to the other pages in your web.

3. When the home page is displayed in Page view, press CTRL+END to place the cursor at the end of the home page.
4. Next, locate the page **background.htm** in the Folder List.

The folders and files in the Folder List are shown in alphabetical order. The icon of each file gives you a clue about what kind of file it is.

You will now drag and drop the Background page onto the bottom of the home page. When you do this, FrontPage will create a hyperlink to the Background page on the home page.

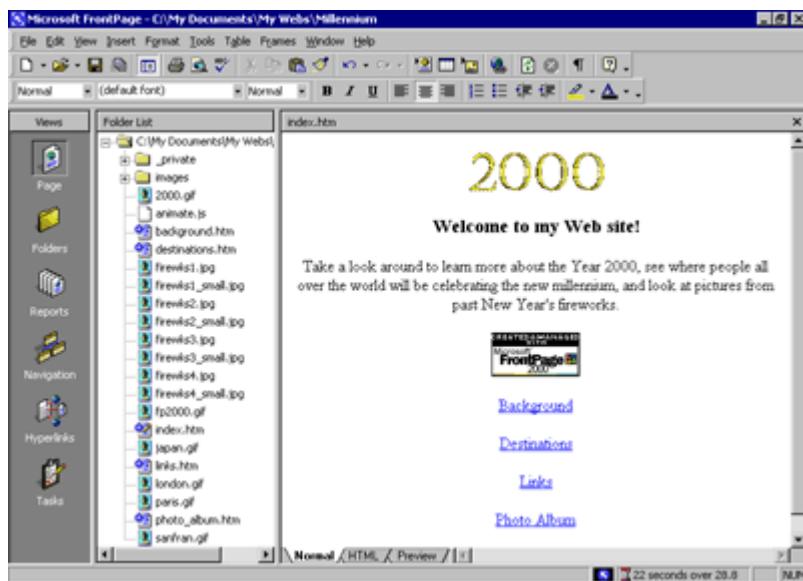
5. Click and hold the mouse button on **background.htm** in the Folder List, move the mouse pointer on the line below the FrontPage button at the bottom of the home page, and then release the mouse button.

FrontPage displays the shortcut mouse pointer while you drag the mouse to indicate that it will not actually insert the Background page, but will create a hyperlink pointing to it.

FrontPage inserts the page title of the Background.htm file ("Background") as the hyperlink text. The blue underlined text shows the presence of the hyperlink.

6. Repeat steps 4 and 5 with the other pages in the Millennium Celebration Web, including Destinations.htm, Links.htm, and Photo\_album.htm. Place each link just after the previous one.
7. On your keyboard, press the DOWN ARROW key to deselect the last hyperlink.

Your page should now look like this:



While you can manually create hyperlinks to the other pages in your web this way, doing so for all pages in a web can become a time-consuming and tedious task, especially for larger webs. Worse, if you decide to add or remove pages in the current web after creating hyperlinks, you'll have to manually add or remove the hyperlinks to them.

FrontPage has a better way to create, manage, and automatically update the navigation hyperlinks that connect your pages together. Before you learn how to do this, let's get rid of the four hyperlinks you just made.

#### ► To use the multiple Undo command

1. On the toolbar, click the small arrow just to the right of the **Undo** button.
2. FrontPage displays the Undo history, which shows the last several actions you can reverse. The first of these actions is selected by default. If you were to click it, then only that action would be reversed. You can also move the mouse over other entries in this list to include them in the **Undo** command.
3. Since we want to get rid of all four hyperlinks you just dragged and dropped onto the home page, move the mouse down the list to select all four occurrences of **Drop**.

The status bar in the Undo history window should read **Undo 4 Actions**.

4. Click the mouse on the last occurrence of **Drop** in the list.

FrontPage reverses the last four actions you took, and the four hyperlinks you created are removed from the home page.

5. To save the current page, click **Save** on the **File** menu, or click the **Save** button on the toolbar.

## Adding Shared Borders and Navigation Bars

For the Millennium Celebration Web, you will let FrontPage manage the hyperlinks that site visitors will click to move around the pages in your web. FrontPage achieves this with a combination of two powerful features: shared borders and automatic navigation bars.

Shared borders are page regions reserved for content that you want to appear consistently throughout the pages in your web. These borders can contain page banners and navigation bars. Page banners display the page title you gave each page when you created or saved it. Navigation bars are a row or column of hyperlinks to the other pages in the current web. FrontPage can automatically update shared borders and navigation bars, so the navigation structure of your web will always work correctly, even when you add, move, or delete pages from the web's structure.

In Lesson 1, you already completed the first step required for automatic navigation bars: creating the basic web structure in Navigation view. Because you have already done this, you'll now enable shared borders throughout your web.

#### ► To create shared borders across a web

1. Click the **Navigation** icon on the Views bar to switch to Navigation view.
2. Click the **Folder List** button to hide the Folder List in this view.
3. On the **Format** menu, click **Shared Borders**.
4. FrontPage displays the **Shared Borders** dialog box. Here, you can specify where on your pages FrontPage should insert shared borders. Because your web structure has two levels of pages — the home page and the pages below it — you will use two kinds of shared borders and two kinds of navigation bars.
5. In the **Shared Borders** dialog box, make sure the **All pages** option is selected.
6. For a horizontal shared border, select the **Top** check box and select the **Include navigation buttons** check box just below it.

7. For a vertical shared border, select the **Left** check box and select the **Include navigation buttons** check box below it.
8. Leave the **Right** and **Bottom** check boxes unchecked, and then click **OK**.

FrontPage creates shared borders and default navigation bars for all the pages in the current web. You'll see what these look like when you return to Page view.

Next, you'll customize the appearance of the default navigation bars. Because they are shared across all pages in the current web, you can change their properties on any page and the change will be reflected across the entire web.

► **To test navigation bar hyperlinks**

1. In Navigation view, double-click the **Home Page**.
2. Click the **Folder List** button to hide the Folder List in Page view.

Note the changes FrontPage has made to the home page. It now contains a top and left shared border. The top border contains a page banner with the name of the current page, and the left border contains a list of navigational hyperlinks that look exactly like the ones you manually created at the beginning of this lesson.

In Page view, you can easily test hyperlinks that point to pages and files in your web.

3. Hold down CTRL and then click the first navigational hyperlink named **Background** on the left side of the page.

FrontPage opens the page the hyperlink points to. On the Background page that is now open, shared borders and navigation bars have also been inserted. On this page, however, the links to the other pages are displayed in the top border, just under the page banner. This is because FrontPage uses the web structure you created in Navigation view to determine the level the current page is on.

By default, the top shared border points to pages on the same level as the current one, whereas the left border points to pages below the current one. In the next section, we'll change this default to another design.

► **To customize navigation bars**

1. On the **Window** menu, click **index.htm**.

FrontPage brings the home page back into view.

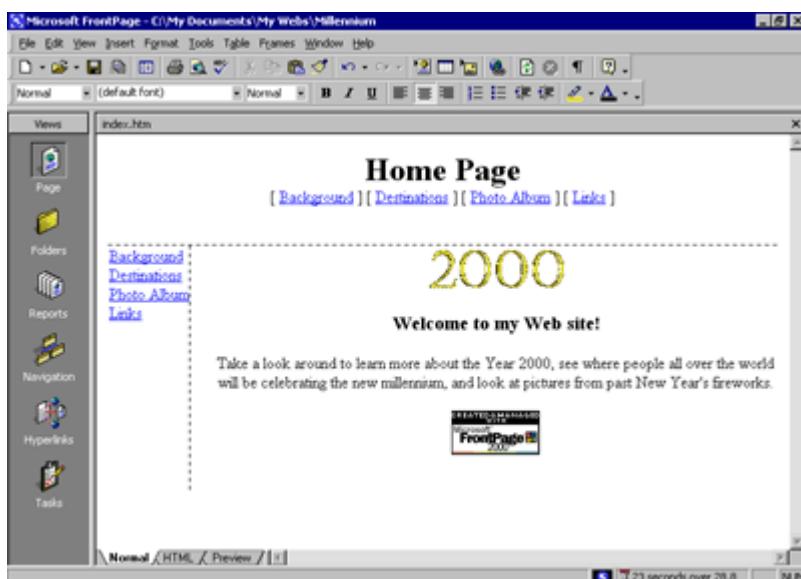
2. In the top border of the home page, double-click the text that reads **Edit the properties for this Navigation Bar to display hyperlinks here**.
3. Double-clicking a navigation bar opens the **Navigation Bar Properties** dialog box. Here, you can customize the appearance of a navigation bar and the hyperlinks it creates.
4. Currently, the horizontal navigation bar is set to link to pages on the same level. Since the home page is on its own level in your navigation structure of your web, there are no other pages on the same level. FrontPage therefore doesn't show any navigation bars in this shared border.

5. For the Millennium Celebration Web, we want to have a horizontal navigation bar on the home page and vertical navigation bars on the other pages. To do this, we'll change the default setup of both navigation bars. You can make these changes on the current page and they'll be reflected throughout your web.
6. In the **Navigation Bar Properties** dialog box, click **Child level**, clear the check boxes for **Home page** and **Parent page**, and then click **OK**.

FrontPage creates a navigation bar with hyperlinks to all the pages below the home page level.

7. Press HOME to deselect the navigation bar.

Your page should now look like this:



Note that the left navigation bar still contains the same set of hyperlinks as the top navigation bar. In the next steps, you'll remove the obvious redundancy, and format the left navigation bar so it is displayed only on the other pages that the home page points to.

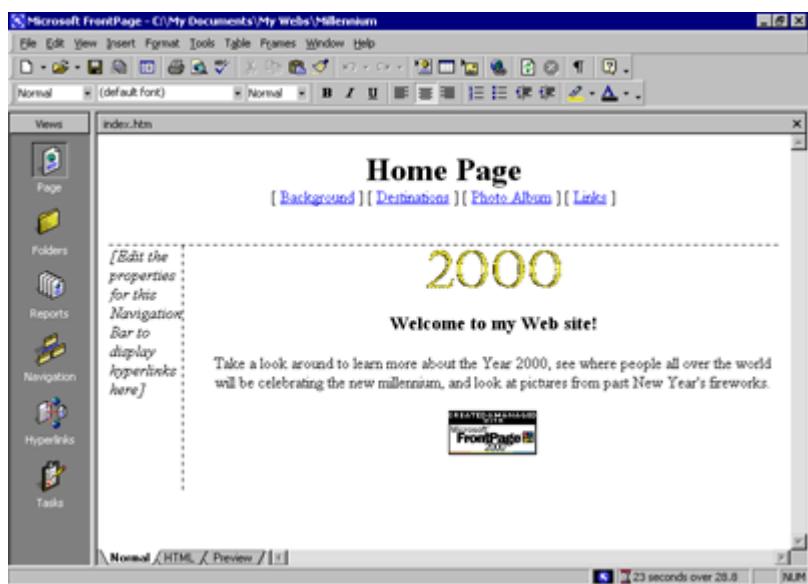
8. In the left border of the home page, double-click the vertical navigation bar.
9. In the **Navigation Bar Properties** dialog box, click **Same level**, select the **Home page** check box, and then click **OK**.

FrontPage changes the navigation bar to the placeholder text that reads "Edit the properties for this Navigation Bar to display hyperlinks here." This text is only shown in Page view while you work; it will not appear in a Web browser.

By pointing the hyperlinks in this navigation bar to the same level as the home page, you are effectively removing the hyperlinks from the left border, because there are no pages on the same level as the home page. This also removes the redundancy between the horizontal and vertical navigation bars.

10. To save the home page, click **Save** on the **File** menu, or click the **Save** button on the toolbar.

Your page should now look like this:



11. On the **Window** menu, click **background.htm**.

FrontPage brings the Background page back into view.

Note that the changes you've made on the home page to both the horizontal and vertical navigation bars are automatically reflected here, as well as on all the other pages in your web.

### Applying a Theme

Although the addition of pictures, lists, forms, shared borders, and navigation bars has given the pages in the Millennium Celebration Web a more streamlined and organized look, you may wonder what to do about the rather bland appearance of black and blue text on a white background. After all, this web is about celebrating an event. You want the pages to look more lively and fun.

Imagine how time-consuming it would be if you had to design a color scheme for text and graphics, and create graphical page banners, navigation buttons, list bullets, and background textures for all the pages in your web. Now imagine how many more custom graphics you would need to create if you maintained more than one Web site and you didn't want any of your webs to look the same.

FrontPage includes more than 50 professionally designed themes with matching color schemes that you can apply to any or all pages in your web. A theme consists of design elements for bullets, fonts, pictures, navigation buttons, and other graphics. When applied, a theme gives pages, page banners, navigation bars, and other elements of a web an attractive and consistent appearance.

#### ► To apply a theme to the Millennium Celebration Web

1. On the **Window** menu, click **index.htm**.

FrontPage brings the home page back into view.

2. On the **Format** menu, click **Theme**.

FrontPage displays the Themes dialog box. Here, you can select from a list of themes that FrontPage installed by default, or choose to install the complete set of themes from your FrontPage 2000 CD-ROM. You can make choices about the appearance of the theme, preview theme elements, and modify the selected theme.

3. Click on some of the different theme names in the scrolling list box.

When you click the name of a theme, the Sample of Theme window shows a sample of the graphical elements that are contained in the selected theme. This way, you can first preview a theme before applying it to selected or all pages in your web.

Before applying a theme, you can select theme options that affect the appearance of the theme's components. For example, selecting Vivid colors applies brighter colors to text and graphics, selecting Active graphics animates certain theme components, and selecting Background picture applies a graphical background to the pages in your web. You can also choose to apply a theme as a cascading style sheet (Apply using CSS).

For the Millennium Celebration Web, you'll clear these defaults.

4. Under **Apply theme to**, make sure **All pages** is selected.
5. In the list of installed themes, click **Artsy**.
6. Clear the check boxes for **Active graphics** and **Background picture**, then click **OK** to apply the theme.

Since this is the first time you're applying a theme to a web, FrontPage displays a message to let you know that applying a theme will overwrite some of the manual formatting you may have done on your pages.

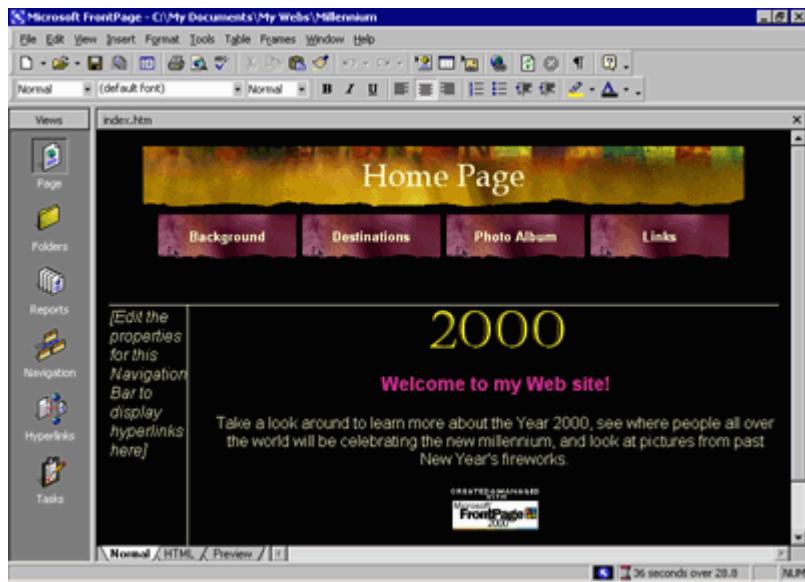
We've purposely not included much manual design work in this tutorial, so you can acknowledge this message and proceed to apply the theme.

7. Click **Yes** to apply the theme.

The theme named "Artsy" is applied to all the pages in your current web.

8. To save the home page, click **Save** on the **File** menu, or click the **Save** button on the toolbar.

Your page should now look like this:



As you can see, applying the theme has dramatically changed the appearance of the home page. The page banner and navigation buttons are no longer plain text; now they're colorful graphics. The page background has changed from white to black, which simulates the night sky that the millennium fireworks will appear in, and the font has changed color and is a little larger.

#### ► **Displaying graphical navigation buttons on all pages**

1. On the **Window** menu, click **background.htm**.

FrontPage brings the Background page back into view.

Note that the page has inherited its theme and theme elements from the home page, but the vertical navigation bar in the left border still shows plain text hyperlinks. By default, vertical navigation bars are displayed as plain text, so they look this way even after you apply a theme. You can easily change navigation bar settings even after a theme is applied.

2. In the left border of the Background page, double-click the vertical navigation bar.
3. Under **Orientation and appearance** in the **Navigation Bar Properties** dialog box, click **Buttons**, and then click **OK**.
4. Click anywhere on the page to deselect the navigation bar.

FrontPage changes the navigation formatting and uses the graphical buttons included with the theme. The web now has an attractive and professional look.

5. To save the page, click **Save** on the **File** menu, or click the **Save** button on the toolbar.

### **Modifying a Theme**

Although the page banner of this theme looks nice, something directly related to the subject matter of the Millennium Celebration Web might fit better. We've prepared a custom page banner that you will use to modify the current theme with. This custom banner provides a colorful fireworks backdrop for the page banner text.

► **To modify a theme**

1. On the **Window** menu, click **index.htm**.

FrontPage brings the home page back into view.

2. On the **Format** menu, click **Theme**.

FrontPage displays the **Themes** dialog box. In the list of themes, the Artsy theme is now the default theme because it has been applied to the current web.

3. In the **Themes** dialog box, make sure **All Pages** is selected.
4. Next, click **Modify**.
5. Under the question **What would you like to modify?** click **Graphics**.

FrontPage displays the **Modify Theme** dialog box. Here, you can supply custom graphics for various theme elements such as page banners, navigation buttons, background pictures, and other elements. FrontPage superimposes text over these graphics, so there is no need to change graphics when you change the names of your pages, or add or remove pages.

For this example, we will change the graphical page banner on which FrontPage places the titles of the pages in the Millennium Celebration Web.

6. In the **Item** list, click **Banner**.
7. On the **Picture** tab, click the **Browse** button below the file name of the current banner graphic.

FrontPage displays the **Select Picture** dialog box and shows the current pictures in your current web. Since the graphical banner we want to use isn't part of the web yet, you will search your file system for it.

8. In the **Select Picture** dialog box, click the **Select File** button.

FrontPage displays the **Select File** dialog box.

9. Navigate to the folder named Program Files\Microsoft Office\Office\Tutorial by double-clicking each folder in this path until the **Look in** box displays the Tutorial folder.

If you downloaded the tutorial files from [www.microsoft.com](http://www.microsoft.com), navigate to the folder named Fptutor\Samples, or to the folder where you placed the files.

10. Double-click the file **2000ban**.

FrontPage replaces the current page banner graphic with the custom graphic.

11. Click **OK** in the **Modify Theme** dialog box, and then click **OK** in the **Themes** dialog box.

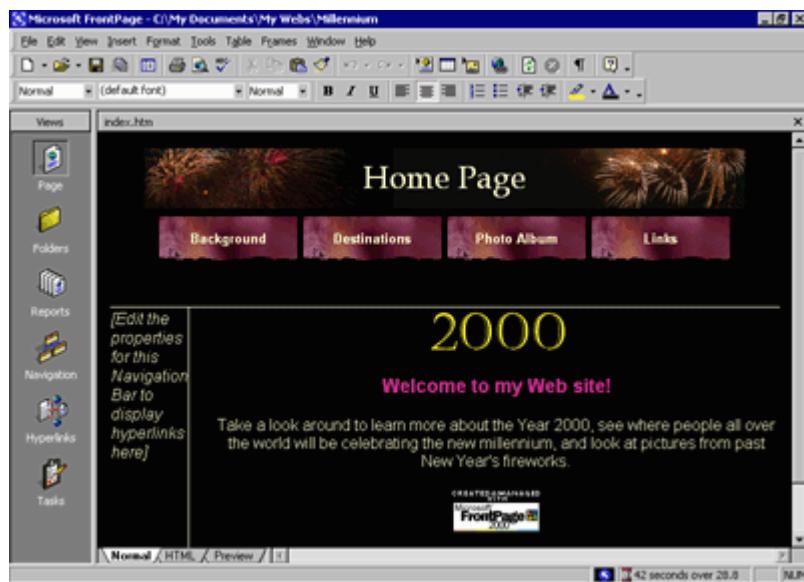
FrontPage displays a message asking you whether you want to save changes to the current theme.

12. Click **Yes**. Enter **Millennium** as the title of the modified theme, and then click **OK**.

FrontPage saves the modified theme and applies the new banner to all pages.

13. To save the page, click **Save** on the **File** menu, or click the **Save** button on the toolbar.

Your page should now look like this:



Congratulations, the Millennium Celebration Web is almost finished! To make sure everything will look great on the World Wide Web, you'll now preview the web in your Web browser.

## Previewing the Millennium Celebration Web

Although Page view shows you the appearance of your Web pages as closely as possible to how they will appear in a Web browser, it displays some page elements and placeholders differently to help you while you design the page. By previewing a page or your entire site in a Web browser before you publish the web, you can make sure that everything looks the way you want it to.

### ► To preview the current web in a Web browser

1. On the **File** menu, click **Preview in Browser**.

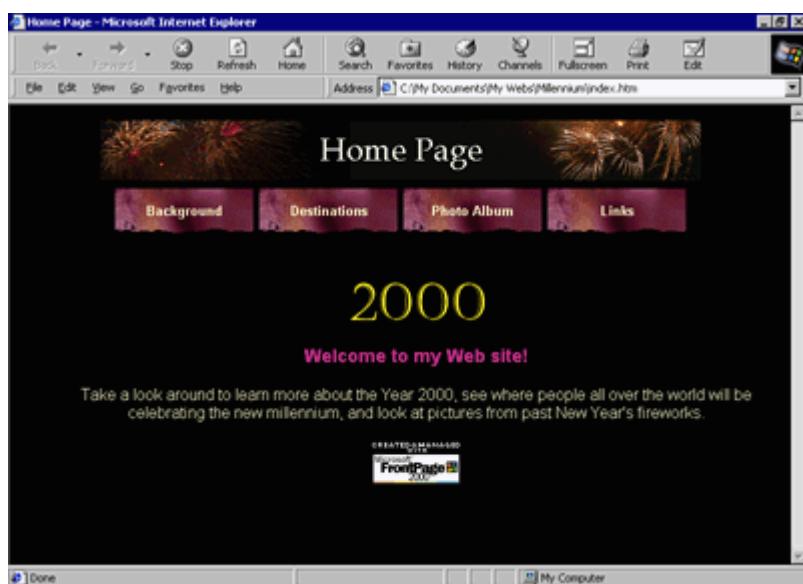
FrontPage displays the **Preview in Browser** dialog box. Here, you can select from the installed Web browsers on your computer, add Web browsers you installed after installing FrontPage, and select the Web browser window size in which you want to preview your web.

The **Preview in Browser** feature loads the current page in your Web browser, so you can see exactly how the page will appear in your favorite Web browser before you publish your web. You must have at least one Web browser installed on your system for this feature to work.

2. In the **Preview in Browser** dialog box, click **Microsoft Internet Explorer**, and then click **Preview**.

FrontPage launches the Web browser and opens the home page.

Your screen should now look like this:



Note that all placeholder text and formatting marks are hidden from view (for example, the empty vertical navigation bar on the left side of the home page does not appear).

3. Click the buttons on the navigation bar to preview some of the other pages.

Note the vertical navigation bar on the pages below the home page. On the Destinations page, scroll down to the feedback form you added to the page. You can enter text in the fields, but the form won't actually work until you publish the Millennium Celebration Web to a Web server.

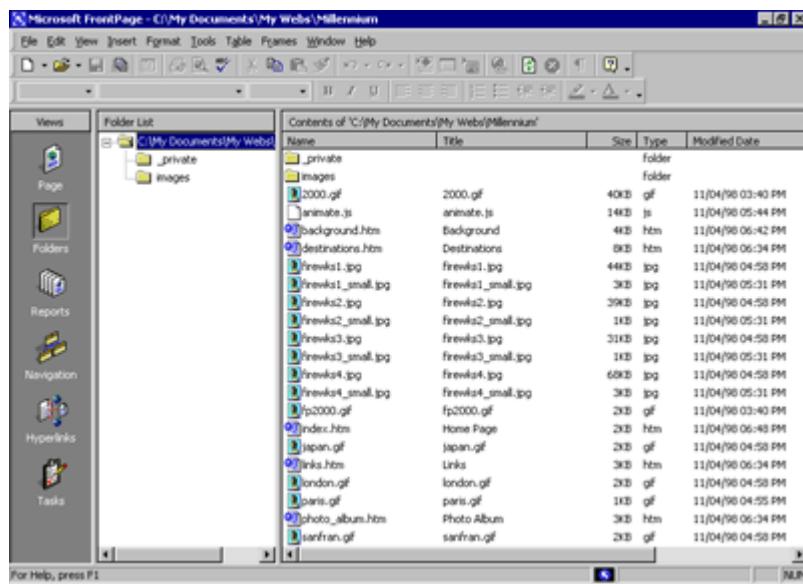
On the Photo Album page, click the picture thumbnails to test the hyperlinks to the full-size pictures. Use your Web browser's Back button to return to the Photo Album page. Finally, on the Links page, note the dynamic animation of the paragraph heading that reads "Links to My Favorite Sites."

4. Close your Web browser when you have finished previewing the Millennium Celebration Web.

## Organizing the Files in your Web

Now that your web contains several pages and files, you will use Folders view to organize them. Similar to Windows Explorer, Folders view lets you manage the files and folders in your web. You can safely rearrange the pages and files in your web without breaking hyperlinks, page banner titles, or navigation button labels.

In Folders view, FrontPage displays a hierarchical list of the folders in your web on the left side of the screen. Clicking on a folder in the Folder List displays its contents on the right side — the Contents pane.



In the following steps, you will move all the picture files in the Millennium Celebration Web to the Images folder FrontPage created as part of the web.

If you were to use Windows Explorer or another file manager to move pages and files from one folder to another, you would break the hyperlinks between your pages and page elements. However, when you maintain your web in Folders view, FrontPage keeps every page and hyperlink in your web updated to keep track of the new locations of files and folders that have been moved.

### ► To move picture files to the Images folder

1. On the Views bar, click the **Folders** icon.

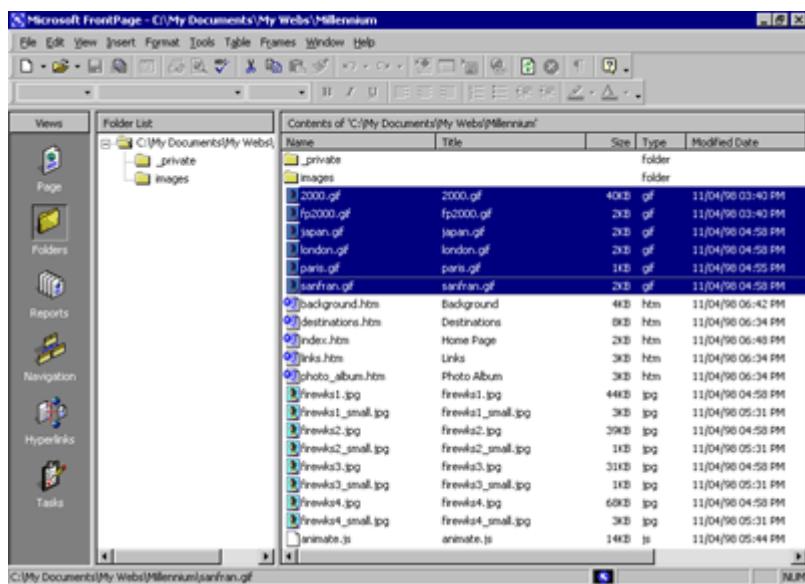
FrontPage switches to Folders view.

2. In the Folder List pane, click the top-level folder labeled **C:\My Documents\My Webs\Millennium**.
3. In the Contents pane, click the **Type** column label.

Clicking on a column label sorts the files in the Contents pane by that criteria. The first time you click a column label, the list is sorted in ascending order; when you click it a second time, it is sorted in descending order.

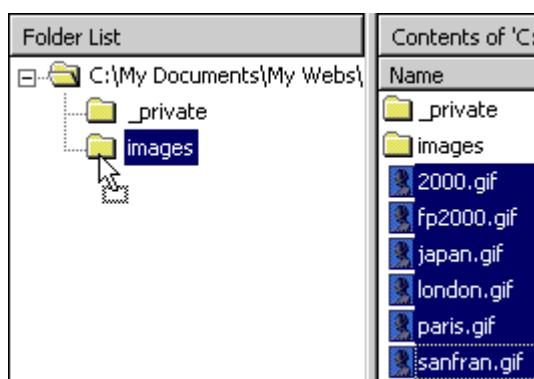
The list of files is now grouped by file type, with all GIF picture files at the top of the list, followed by HTM files (pages) in the middle, and all JPG pictures at the bottom of the list.

4. In the Contents pane, click the first picture file (**2000.gif**) at the top of the list to select it.
5. Next, while holding down SHIFT, click the last GIF picture file in the list (**sanfran.gif**).



In Folders view, FrontPage supports all standard Windows selection shortcuts, such as SHIFT+CLICK for selecting ranges of files, and CTRL+CLICK for selecting noncontiguous files.

6. Click and hold the right mouse button while the pointer is over any of the selected GIF file icons.
7. Next, drag the mouse pointer over to the Images folder in the Folder List pane.



8. When the Images folder is selected, release the mouse button and click **Move Here** on the shortcut menu.

FrontPage displays the **Rename** dialog box while it is moving the selected GIF image files to the Images folder because it is automatically updating all hyperlinks to these files in the current web.

9. Repeat steps 4 through 8 with all JPG picture files, starting with Firewks1.jpg and ending with Firewks4\_small.jpg.
10. In the Contents pane, click the **Name** column label to arrange the remaining list of folders and files by their name again.

You've successfully grouped all picture files in the Images folder.

When you work with your own webs, you can group sound files, movie clips, and other types of files in their own folders. You can create new folders in Folders view as needed and delete the ones you no longer need.

#### ► To create a new folder

1. In the **Folder List**, click the folder in which you want to create a new subfolder.

Folders can be expanded and collapsed in the **Folder List** to bring their subfolders into view. Click the plus (+) and minus (-) signs next to a folder's name to display or hide its subfolders.

2. On the **File** menu, point to **New** and then click **Folder**.

FrontPage creates a new folder with a temporary name.

3. When the folder's temporary name (**New\_Folder**) is selected, type a new name for the folder, then press ENTER.

The new folder is renamed, and you can now drag and drop files into it.

For this tutorial, we don't need the extra folder you just created, so you will delete it before we get ready to publish the web.

4. In the **Folder List**, right-click the folder you just created.
5. On the shortcut menu, click **Delete**.
6. In the **Confirm Delete** dialog box, click **Yes**.

FrontPage removes the folder from the web.

## Generating a Site Summary

Reports view is an important tool that shows you the overall health and condition of your web before you publish it to the World Wide Web. You can generate custom reports about your web in up to 14 categories.

#### ► To generate a Site Summary report

- On the Views bar, click the **Reports** icon.

FrontPage switches to Reports view. The default report is the Site Summary. This report shows you the overall statistics of the pages and files in the Millennium Celebration Web. Here are some important ones to look at before you publish your web:

- **All files:** you currently have 21 files in your web, totaling approximately 275K in size. This is the amount of space you'll need to have available on the Web server that will host your web.
- **Slow pages:** this category shows pages that are slow to download at the targeted download speed. Because you created small thumbnails of the large fireworks pictures in your online Photo Album, your web currently doesn't have any slow pages for you to worry about.
- **Broken hyperlinks:** If any broken hyperlinks are reported here, double-click the **Broken hyperlinks** row to view details about this category. FrontPage lists unverified hyperlinks, such as the external hyperlinks on your Links page, and links that are broken and do not work.
- You can verify that a hyperlink still points to an active Web site by right-clicking the link in Reports view and choosing **Verify** from the shortcut menu. To fix a broken hyperlink, you must open the page it is on and repair the URL the hyperlink points to.

## Spell Checking

While you can use automatic background spell-checking and per-page spell-checking in Page view, using the Spelling command in any web view lets you check the spelling of all (or selected) pages across the current web.

You can check the spelling of page elements that can be edited directly on the page. Other text, such as page titles in page banners or text contained in FrontPage-based components, are not included in the spelling check.

### ► To check spelling in the current web

1. On the **Tools** menu, click **Spelling**.

FrontPage displays the **Spelling** dialog box. Here, you can specify whether FrontPage should check the spelling of selected pages only, or of the entire web.

2. In the **Spelling** dialog box, click **Entire web**, and then select the **Add a task for each page with misspellings** check box.

FrontPage will add a task to the Tasks list for each page on which misspelled text is found. You will learn about Tasks view in the next section.

3. In the **Spelling** dialog box, click **Start** to begin the spelling check.

FrontPage expands the **Spelling** dialog box to display the progress of the spelling check.

When the operation has been completed, FrontPage displays the misspelled words and the number of tasks that were added to the Tasks list in Tasks view.

4. Click **Cancel** to dismiss the **Spelling** dialog box.

The spelling check is complete, but the corrections will not be made until you complete the tasks in the Tasks list.

## Replacing Text on Pages

The Replace command makes it easy to find and replace content on selected pages or all pages in the current web. While you can use the command to replace text on the current page in Page view, using it in any web view lets you replace text in all (or selected) pages across the current web.

You can replace any text that can be edited directly on the page. Other text, such as page titles in page banners or text contained in FrontPage-based components, cannot be automatically replaced.

### ► To replace text on all pages in the current web

1. On the **Edit** menu, click **Replace**.
2. FrontPage displays the **Replace** dialog box. Here, you enter the string of text to be found and what you want to replace it with. You can choose to replace text on all pages in the current web, or on selected pages only.
3. In the **Replace** dialog box, type **Welcome to my Web site** in the **Find what** box.
4. In the **Replace with** box, type **Thanks for visiting my Web site**.
5. Click the **Match case** check box, and then click **Find in Web**.

FrontPage expands the **Replace** dialog box to display the progress of the search. The search text you want to replace is found on the home page, Index.htm.

When the operation has been completed, FrontPage displays the number of occurrences it has found.

6. Click **Add Task** in the **Replace** dialog box.
7. Click **Cancel** to dismiss the **Replace** dialog box.

The replacement search is complete, but the actual replacement will not occur until you complete the task in Tasks view.

## Completing Web Tasks

Tasks view displays the list of all outstanding tasks associated with the current web. Tasks are items that need your attention before you publish the web.

In the previous exercises, you added tasks to a list when you deferred certain actions. For example, when you checked the spelling of the pages in your web, you chose to add a new task for each page containing misspellings. By adding tasks to the list, you can complete such corrections all at once.

If you are working in a web development environment or on an intranet, Tasks view makes it easy to track web tasks and assign them to other authors who work on the same web.

## To complete tasks in Tasks view

1. On the **Views** bar, click the **Tasks** icon.

FrontPage displays the Tasks list.

2. Double-click the first task on the list, labeled "Fix misspelled words."
3. FrontPage displays the **Task Details** dialog box. Here, you can see details about the task you're selected. You can set the priority of the task, assign it to another author on your network, or complete the task and remove it from the list.
4. In the **Task Details** dialog box, click **Start Task**.

FrontPage switches to Page view and opens the page containing the misspelled words.

5. In the **Spelling** dialog box, click **Ignore** when FrontPage questions the name "Balleny."
6. Click **Add** to add "Cheops," the name of the Egyptian king, to your dictionary.

FrontPage shares custom dictionaries with other Microsoft Office applications, so you don't need to add custom words in each application separately.

When you add verified words to your dictionary, they will not be questioned again.

7. Click **OK**.

FrontPage completes the spelling check. If you want to, you can now return to Tasks view and mark this task as completed.

Although it is not required that you complete every task before publishing your web, it is a good idea to review this list when you are finished making changes to the web. Tasks view helps you manage webs by flagging important reminders for you.

## Publishing the Millennium Celebration Web

When you publish your web on the World Wide Web — or your company intranet — FrontPage automatically verifies your hyperlinks, the addresses of your pages, and the paths to your files.

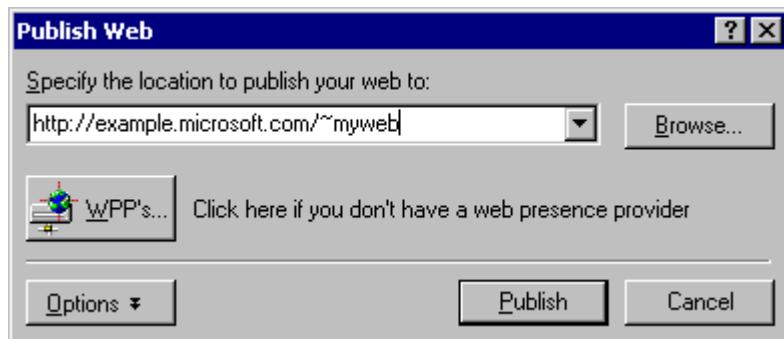
**Note:** If you do not want to publish the Millennium Celebration Web to your Web server, read this procedure for reference only, without actually completing the steps.

### To publish the Millennium Celebration Web

1. Close all open pages in Page view.
2. On the **File** menu, click **Publish Web**, or click the **Publish Web** button on the toolbar.

FrontPage displays the Publish Web dialog box. Here, you specify the location on the World Wide Web or your corporate intranet to which you want to publish your web. Your Internet service provider can tell you this information.

You need Internet access through an Internet service provider before you can publish your web to the World Wide Web. If you want to sign on with a Web Presence Provider that can host FrontPage-enabled webs, click the WPPs button in the Publish Web dialog box.



3. In the **Publish Web** dialog box, enter the URL of your target Web server, (such as <http://example.microsoft.com/~myweb>), and then click **Publish**.

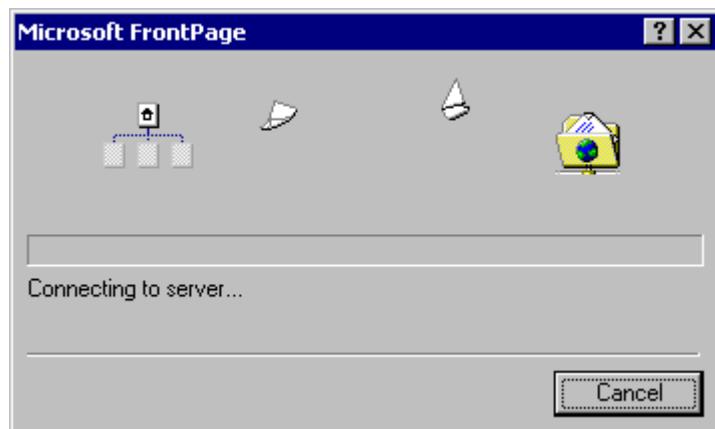
FrontPage publishes the current web from your computer to the World Wide Web or intranet Web server you specified.

If FrontPage detects that you are publishing to a Web server that does not support the FrontPage Server Extensions, it will publish the current web via the FTP file transfer protocol.

If the Web server to which you are publishing your webs has the FrontPage Server Extensions installed, your webs will have full functionality of FrontPage-based components and Web scripts that you may have inserted on your pages.

Publishing webs to a Web server that does not have the FrontPage Server Extensions installed may disable some functionality contained on your pages, such as the feedback form you added. FrontPage will display informational messages during the publication process to alert you of such conditions.

During the publishing process, FrontPage displays a progress bar to indicate how much time is required to transfer your web to the target Web server.



The speed at which FrontPage publishes your web depends on your connection speed, as well as the number and complexity of the pages and files in your web. When FrontPage has successfully published your web, it provides a hyperlink to your new Web site in the confirmation dialog box. Click the link to open the published Web site in your Web browser.you have successfully completed the FrontPage Tutorial. You are now ready to create and publish your very own FrontPage-based webs

# Cascading Style Sheets (CSS)

## What is CSS?

- **CSS** stands for Cascading Style Sheets
- 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).

## 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.

## 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  
}
```

## 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 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.

Property	Description	Values	IE	F	N
background	A shorthand property for setting all background properties in one declaration	<i>background-color</i> <i>background-image</i> <i>background-repeat</i> <i>background-attachment</i> <i>background-position</i>	4	1	6
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	scroll fixed	4	1	6
background-color	Sets the background color of an element	<i>color-rgb</i> <i>color-hex</i> <i>color-name</i> transparent	4	1	4
background-image	Sets an image as the background	<i>url</i> none	4	1	4
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 <i>x-% y-%</i> <i>x-pos y-pos</i>	4	1	6
background-repeat	Sets if/how a background image will be repeated	repeat repeat-x repeat-y no-repeat	4	1	4

## 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.

Property	Description	Values	IE	F	N
color	Sets the color of a text	<i>color</i>	3	1	4
direction	Sets the text direction	ltr rtl	6	1	6
letter-spacing	Increase or decrease the space between characters	normal <i>length</i>	4	1	6
text-align	Aligns the text in an element	left right center justify	4	1	4
text-decoration	Adds decoration to text	none underline overline line-through blink	4	1	4
text-indent	Indents the first line of text in an element	<i>length</i> %	4	1	4
text-shadow		none <i>color</i> <i>length</i>			
text-transform	Controls the letters in an element	none capitalize uppercase lowercase	4	1	4
unicode-bidi		normal embed bidi-override	5		
white-space	Sets how white space inside an element is handled	normal pre nowrap	5	1	4
word-spacing	Increase or decrease the space between words	normal <i>length</i>	6	1	6

## 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.

Property	Description	Values	IE	F	N
font	A shorthand property for setting all of the properties for a font in one declaration	<i>font-style</i> <i>font-variant</i> <i>font-weight</i> <i>font-size/line-height</i> <i>font-family</i> caption icon menu message-box small-caption status-bar	4	1	4
font-family	A prioritized list of font family names and/or generic family names for an element	<i>family-name</i> <i>generic-family</i>	3	1	4
font-size	Sets the size of a font	xx-small x-small small	3	1	4

		medium large x-large xx-large smaller larger <i>length</i> %			
<u>font-size-adjust</u>	Specifies an aspect value for an element that will preserve the x-height of the first-choice font	none <i>number</i>	-	-	-
<u>font-stretch</u>	Condenses or expands the current font-family	normal wider narrower ultra-condensed extra-condensed condensed semi-condensed semi-expanded expanded extra-expanded ultra-expanded	-	-	-
<u>font-style</u>	Sets the style of the font	normal italic oblique	4	1	4
<u>font-variant</u>	Displays text in a small-caps font or a normal font	normal small-caps	4	1	6
<u>font-weight</u>	Sets the weight of a font	normal bold bolder lighter 100 200 300 400 500 600 700 800 900	4	1	4

## 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.

Property	Description	Values	IE	F	N
<u>border</u>	A shorthand property for setting all of the properties for the four borders in one declaration	<i>border-width</i> <i>border-style</i> <i>border-color</i>	4	1	4
<u>border-bottom</u>	A shorthand property for setting all of the properties for the bottom border in one declaration	<i>border-bottom-width</i> <i>border-style</i> <i>border-color</i>	4	1	6
<u>border-bottom-color</u>	Sets the color of the bottom border	<i>border-color</i>	4	1	6

<u>border-bottom-style</u>	Sets the style of the bottom border	<i>border-style</i>	4	1	6
<u>border-bottom-width</u>	Sets the width of the bottom border	thin medium thick <i>length</i>	4	1	4
<u>border-color</u>	Sets the color of the four borders, can have from one to four colors	<i>color</i>	4	1	6
<u>border-left</u>	A shorthand property for setting all of the properties for the left border in one declaration	<i>border-left-width</i> <i>border-style</i> <i>border-color</i>	4	1	6
<u>border-left-color</u>	Sets the color of the left border	<i>border-color</i>	4	1	6
<u>border-left-style</u>	Sets the style of the left border	<i>border-style</i>	4	1	6
<u>border-left-width</u>	Sets the width of the left border	thin medium thick <i>length</i>	4	1	4
<u>border-right</u>	A shorthand property for setting all of the properties for the right border in one declaration	<i>border-right-width</i> <i>border-style</i> <i>border-color</i>	4	1	6
<u>border-right-color</u>	Sets the color of the right border	<i>border-color</i>	4	1	6
<u>border-right-style</u>	Sets the style of the right border	<i>border-style</i>	4	1	6
<u>border-right-width</u>	Sets the width of the right border	thin medium thick <i>length</i>	4	1	4
<u>border-style</u>	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
<u>border-top</u>	A shorthand property for setting all of the properties for the top border in one declaration	<i>border-top-width</i> <i>border-style</i> <i>border-color</i>	4	1	6
<u>border-top-color</u>	Sets the color of the top border	<i>border-color</i>	4	1	6
<u>border-top-style</u>	Sets the style of the top border	<i>border-style</i>	4	1	6
<u>border-top-width</u>	Sets the width of the top border	thin medium thick <i>length</i>	4	1	4
<u>border-width</u>	A shorthand property for setting the width of the four borders in one declaration, can have from one to four values	thin medium thick <i>length</i>	4	1	4

## 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.

Property	Description	Values	IE	F	N
<u>margin</u>	A shorthand property for setting the margin properties in one declaration	<i>margin-top</i> <i>margin-right</i> <i>margin-bottom</i> <i>margin-left</i>	4	1	4
<u>margin-bottom</u>	Sets the bottom margin of an element	auto <i>length</i> %	4	1	4
<u>margin-left</u>	Sets the left margin of an element	auto <i>length</i> %	3	1	4
<u>margin-right</u>	Sets the right margin of an element	auto <i>length</i> %	3	1	4
<u>margin-top</u>	Sets the top margin of an element	auto <i>length</i> %	3	1	4

## 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.

Property	Description	Values	IE	F	N
<u>padding</u>	A shorthand property for setting all of the padding properties in one declaration	<i>padding-top</i> <i>padding-right</i> <i>padding-bottom</i> <i>padding-left</i>	4	1	4
<u>padding-bottom</u>	Sets the bottom padding of an element	<i>length</i> %	4	1	4
<u>padding-left</u>	Sets the left padding of an element	<i>length</i> %	4	1	4
<u>padding-right</u>	Sets the right padding of an element	<i>length</i> %	4	1	4
<u>padding-top</u>	Sets the top padding of an element	<i>length</i> %	4	1	4

## 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.

Property	Description	Values	IE	F	N
<u>list-style</u>	A shorthand property for setting all of the properties for a list in one declaration	<i>list-style-type</i> <i>list-style-position</i> <i>list-style-image</i>	4	1	6

<u>list-style-image</u>	Sets an image as the list-item marker	none <i>url</i>	4	1	6
<u>list-style-position</u>	Sets where the list-item marker is placed in the list	inside outside	4	1	6
<u>list-style-type</u>	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
marker-offset		auto <i>length</i>	1	7	

## 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.

Property	Description	Values	IE	F	N
<u>height</u>	Sets the height of an element	auto <i>length</i> %	4	1	6
<u>line-height</u>	Sets the distance between lines	normal <i>number</i> <i>length</i> %	4	1	4
<u>max-height</u>	Sets the maximum height of an element	none <i>length</i> %	-	1	6
<u>max-width</u>	Sets the maximum width of an element	none <i>length</i> %	-	1	6
<u>min-height</u>	Sets the minimum height of an element	<i>length</i> %	-	1	6
<u>min-width</u>	Sets the minimum width of an element	<i>length</i> %	-	1	6
<u>width</u>	Sets the width of an element	auto % <i>length</i>	4	1	4

## 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.

Property	Description	Values	IE	F	N
<u>clear</u>	Sets the sides of an element where other floating elements are not allowed	left right both none	4	1	4
<u>cursor</u>	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
<u>display</u>	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
<u>float</u>	Sets where an image or a text will appear in another element	left right none	4	1	4
<u>position</u>	Places an element in a static, relative, absolute or fixed position	static relative absolute fixed	4	1	4
<u>visibility</u>	Sets if an element should be visible or invisible	visible hidden collapse	4	1	6

## 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.

Property	Description	Values	IE	F	N
<u>bottom</u>	Sets how far the bottom edge of an element is above/below the bottom edge of the parent element	auto % <i>length</i>	5	1	6
<u>clip</u>	Sets the shape of an element. The element is clipped into this shape, and displayed	<i>shape</i> auto	4	1	6
<u>left</u>	Sets how far the left edge of an element is to the right/left of the left edge of the parent element	auto % <i>length</i>	4	1	4
<u>overflow</u>	Sets what happens if the content of an element overflow its area	visible hidden scroll auto	4	1	6
<u>position</u>	Places an element in a static, relative, absolute or fixed position	static relative absolute fixed	4	1	4
<u>right</u>	Sets how far the right edge of an element is to the left/right of the right edge of the parent element	auto % <i>length</i>	5	1	6
<u>top</u>	Sets how far the top edge of an element is above/below the top edge of the parent element	auto % <i>length</i>	4	1	4
<u>vertical-align</u>	Sets the vertical alignment of an element	baseline sub super top text-top middle bottom text-bottom <i>length</i> %	4	1	4
<u>z-index</u>	Sets the stack order of an element	auto <i>number</i>	4	1	6

## 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.

In this example, the selector matches **any p element that is the first child of a div element**, and indents the first paragraph inside a div element:

```
div > p:first-child
{
text-indent:25px
}
```

This selector will match the first paragraph inside the div in the following HTML:

```
<div>
<p>
First paragraph in div.
This paragraph will be indented.
</p>
<p>
Second paragraph in div.
This paragraph will not be indented.
</p>
</div>
```

but it will not match the paragraph in this HTML:

```
<div>
<h1>Header</h1>
<p>
The first paragraph inside the div.
This paragraph will not be indented.
</p>
</div>
```

In this example, the selector matches **any em element that is the first child of a p element**, and sets the font-weight to bold for the first em inside a p element:

```
p:first-child em
{
font-weight:bold
}
```

For example, the em in the HTML below is the first child of the paragraph:

```
<p>I am a <em>strong</em> man.</p>
```

In this example, the selector matches **any a element that is the first child of any element**, and sets the text-decoration to none:

```
a:first-child
{
text-decoration:none
}
```

For example, the first a in the HTML below is the first child of the paragraph and will not be underlined. But the second a in the paragraph is not the first child of the paragraph and will be underlined:

```
<p>
Visit <a href="http://www.w3schools.com">W3Schools</a>
and learn CSS!
Visit <a href="http://www.w3schools.com">W3Schools</a>
and learn HTML!
</p>
```

## 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.

Pseudo-class	Purpose	IE	F	N
:active	Adds special style to an activated element	4	1	8
:focus	Adds special style to an element while the element has	-	-	-

	focus			
:hover	Adds special style to an element when you mouse over it	4	1	7
:link	Adds special style to an unvisited link	3	1	4
:visited	Adds special style to a visited link	3	1	4
:first-child	Adds special style to an element that is the first child of some other element		1	7
:lang	Allows the author to specify a language to use in a specified element		1	8

## 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 {font-size: 12pt}
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 {font-size: 12pt}
p:first-letter {font-size: 200%; float: left}
```

```
<p>The first words of an article.</p>
```

The output could be something like this:

```
| he 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 {font-size: 12pt}  
p:first-letter {color: #FF0000; font-size: 200%}  
p:first-line {color: #0000FF}  
<p>The first words of an article</p>
```

The output could be something like this:

```
| he 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 an element.

The style below will play a sound before each occurrence of a header one element.

```
h1:before
{
content: url(beep.wav)
}
```

## CSS2 - The :after Pseudo-element

The ":after" pseudo-element can be used to insert some content after an element.

The style below will play a sound after each occurrence of a header one 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 an element		1.5	8	2
:after	Inserts some content after an element		1.5	8	2

## 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 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.

# XML

## What is XML?

- XML stands for EXtensible Markup Language
- XML is a **markup language** much like HTML
- XML was designed to **describe data**
- XML tags are not predefined. You must **define your own tags**
- XML uses a **Document Type Definition** (DTD) or an **XML Schema** to describe the data
- XML with a DTD or XML Schema is designed to be **self-descriptive**
- XML is a W3C Recommendation

## The Main Difference Between XML and HTML:

XML was designed to carry data. XML is not a replacement for HTML.  
XML and HTML were designed with different goals:

XML was designed to describe data and to focus on what data is.  
HTML was designed to display data and to focus on how data looks.

HTML is about displaying information, while XML is about describing information.

## XML Does not DO Anything

XML was not designed to DO anything. Maybe it is a little hard to understand, but XML does not DO anything. XML was created to structure, store and to send information.

The following example is a note to Tove from Jani, stored as XML:

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The note has a header and a message body. It also has sender and receiver information. But still, this XML document does not DO anything. It is just pure information wrapped in XML tags. Someone must write a piece of software to send, receive or display it.

## XML is Free and Extensible

XML tags are not predefined. You must "invent" your own tags. The tags used to mark up HTML documents and the structure of HTML documents are predefined. The author of HTML documents can only use tags that are defined in the HTML standard (like `<p>`, `<h1>`, etc.).

XML allows the author to define his own tags and his own document structure.

The tags in the example above (like `<to>` and `<from>`) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

## **XML is a Complement to HTML**

XML is not a replacement for HTML. It is important to understand that XML is not a replacement for HTML. In future Web development it is most likely that XML will be used to describe the data, while HTML will be used to format and display the same data.

My best description of XML is this: **XML is a cross-platform, software and hardware independent tool for transmitting information.**

## **XML in Future Web Development**

XML is going to be everywhere. We have been participating in XML development since its creation. It has been amazing to see how quickly the XML standard has been developed and how quickly a large number of software vendors have adopted the standard.

We strongly believe that XML will be as important to the future of the Web as HTML has been to the foundation of the Web and that XML will be the most common tool for all data manipulation and data transmission.

## **XML can Separate Data from HTML**

With XML, your data is stored outside your HTML. When HTML is used to display data, the data is stored inside your HTML. With XML, data can be stored in separate XML files. This way you can concentrate on using HTML for data layout and display, and be sure that changes in the underlying data will not require any changes to your HTML.

XML data can also be stored inside HTML pages as "Data Islands". You can still concentrate on using HTML only for formatting and displaying the data.

## **XML is Used to Exchange Data**

With XML, data can be exchanged between incompatible systems. In the real world, computer systems and databases contain data in incompatible formats. One of the most time-consuming challenges for developers has been to exchange data between such systems over the Internet.

Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications.

## **XML and B2B**

With XML, financial information can be exchanged over the Internet. Expect to see a lot about XML and B2B (Business To Business) in the near future. XML is going to be the main language for exchanging financial information between businesses over the Internet. A lot of interesting B2B applications are under development.

## **XML Can be Used to Share Data**

With XML, plain text files can be used to share data. Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data. This makes it much easier to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers.

## **XML Can be Used to Store Data**

With XML, plain text files can be used to store data. XML can also be used to store data in files or in databases. Applications can be written to store and retrieve information from the store, and generic applications can be used to display the data.

## **XML Can Make your Data More Useful**

With XML, your data is available to more users. Since XML is independent of hardware, software and application, you can make your data available to other than only standard HTML browsers.

Other clients and applications can access your XML files as data sources, like they are accessing databases. Your data can be made available to all kinds of "reading machines" (agents), and it is easier to make your data available for blind people, or people with other disabilities.

## **XML Can be Used to Create New Languages**

XML is the mother of WAP and WML. The Wireless Markup Language (WML), used to markup Internet applications for handheld devices like mobile phones, is written in XML . The syntax rules of XML are very simple and very strict. The rules are very easy to learn, and very easy to use. Because of this, creating software that can read and manipulate XML is very easy.

### **An Example XML Document**

XML documents use a self-describing and simple syntax.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The first line in the document - the XML declaration - defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set.

The next line describes the root element of the document (like it was saying: "this document is a note"):

```
<note>
```

The next 4 lines describe 4 child elements of the root (to, from, heading, and body):

```
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
```

And finally the last line defines the end of the root element:

```
</note>
```

Can you detect from this example that the XML document contains a Note to Tove from Jani? Don't you agree that XML is pretty self-descriptive?

## **All XML Elements Must Have a Closing Tag**

With XML, it is illegal to omit the closing tag. In HTML some elements do not have to have a closing tag. The following code is legal in HTML:

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

In XML all elements must have a closing tag, like this:

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

**Note:** You might have noticed from the previous example that the XML declaration did not have a closing tag. This is not an error. The declaration is not a part of the XML document itself. It is not an XML element, and it should not have a closing tag.

### XML Tags are Case Sensitive

Unlike HTML, XML tags are case sensitive. With XML, the tag `<Letter>` is different from the tag `<letter>`.

Opening and closing tags must therefore be written with the same case:

```
<Message>This is incorrect</message>  
<message>This is correct</message>
```

---

### XML Elements Must be Properly Nested

Improper nesting of tags makes no sense to XML. In HTML some elements can be improperly nested within each other like this:

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

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

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

### XML Documents Must Have a Root Element

All XML documents must contain a single tag pair to define a root element. All other elements must be within this root element. All elements can have sub elements (child elements). Sub elements must be correctly nested within their parent element:

```
<root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>
```

### XML Attribute Values Must be Quoted

With XML, it is illegal to omit quotation marks around attribute values. XML elements can have attributes in name/value pairs just like in HTML. In XML the attribute value must always be quoted. Study the two XML documents below. The first one is incorrect, the second is correct:

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<note date=12/11/2002>
```

```
<to>Tove</to>
<from>Jani</from>
</note>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note date="12/11/2002">
<to>Tove</to>
<from>Jani</from>
</note>
```

The error in the first document is that the date attribute in the note element is not quoted.

This is correct: date="12/11/2002". This is incorrect: date=12/11/2002.

## **With XML, White Space is Preserved**

With XML, the white space in your document is not truncated. This is unlike HTML. With HTML, a sentence like this:

Hello my name is Tove,

will be displayed like this:

Hello my name is Tove,

because HTML reduces multiple, consecutive white space characters to a single white space.

## **With XML, CR / LF is Converted to LF**

With XML, a new line is always stored as LF. Do you know what a typewriter is? Well, a typewriter is a mechanical device which was used last century to produce printed documents. :-)

After you have typed one line of text on a typewriter, you have to manually return the printing carriage to the left margin position and manually feed the paper up one line.

In Windows applications, a new line is normally stored as a pair of characters: carriage return (CR) and line feed (LF). The character pair bears some resemblance to the typewriter actions of setting a new line. In Unix applications, a new line is normally stored as a LF character. Macintosh applications use only a CR character to store a new line.

## **Comments in XML**

The syntax for writing comments in XML is similar to that of HTML.

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

There is nothing special about XML. It is just plain text with the addition of some XML tags enclosed in angle brackets.

Software that can handle plain text can also handle XML. In a simple text editor, the XML tags will be visible and will not be handled specially.

In an XML-aware application however, the XML tags can be handled specially. The tags may or may not be visible, or have a functional meaning, depending on the nature of the application.

## **XML Elements are extensible and they have relationships.**

XML Elements have simple naming rules.

## XML Elements are Extensible

XML documents can be extended to carry more information. Look at the following XML NOTE example:

```
<note>
<to>Tove</to>
<from>Jani</from>
<body>Don't forget me this weekend!</body>
</note>
```

Let's imagine that we created an application that extracted the `<to>`, `<from>`, and `<body>` elements from the XML document to produce this output:

### MESSAGE

**To:** Tove  
**From:** Jani

Don't forget me this weekend!

Imagine that the author of the XML document added some extra information to it:

```
<note>
<date>2002-08-01</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Should the application break or crash?

No. The application should still be able to find the `<to>`, `<from>`, and `<body>` elements in the XML document and produce the same output.

## XML documents are Extensible.

XML Elements have Relationships Elements are related as parents and children. To understand XML terminology, you have to know how relationships between XML elements are named, and how element content is described.

Imagine that this is a description of a book:

### My First XML

#### Introduction to XML

- What is HTML
- What is XML

#### XML Syntax

- Elements must have a closing tag
- Elements must be properly nested

Imagine that this XML document describes the book:

```

<book>
<title>My First XML</title>
<prod id="33-657" media="paper"></prod>
<chapter>Introduction to XML
<para>What is HTML</para>
<para>What is XML</para>
</chapter>

<chapter>XML Syntax
<para>Elements must have a closing tag</para>
<para>Elements must be properly nested</para>
</chapter>

</book>

```

Book is the **root element**. Title, prod, and chapter are **child elements** of book. Book is the **parent element** of title, prod, and chapter. Title, prod, and chapter are **siblings** (or **sister elements**) because they have the same parent.

## Elements have Content

Elements can have different content types. An **XML element** is everything from (including) the element's start tag to (including) the element's end tag.

An element can have **element content**, **mixed content**, **simple content**, or **empty content**. An element can also have **attributes**.

In the example above, book has **element content**, because it contains other elements. Chapter has **mixed content** because it contains both text and other elements. Para has **simple content** (or **text content**) because it contains only text. Prod has **empty content**, because it carries no information.

In the example above only the prod element has **attributes**. The **attribute** named id has the **value** "33-657". The **attribute** named media has the **value** "paper".

## Element Naming

XML elements must follow these naming rules:

- Names can contain letters, numbers, and other characters
- Names must not start with a number or punctuation character
- Names must not start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces

Take care when you "invent" element names and follow these simple rules:

Any name can be used, no words are reserved, but the idea is to make names descriptive. Names with an underscore separator are nice.

Examples: <first\_name>, <last\_name>.

Avoid "-" and "." in names. For example, if you name something "first-name," it could be a mess if your software tries to subtract name from first. Or if you name something "first.name," your software may think that "name" is a property of the object "first."

Element names can be as long as you like, but don't exaggerate. Names should be short and simple, like this: <book\_title> not like this: <the\_title\_of\_the\_book>.

XML documents often have a corresponding database, in which fields exist corresponding to elements in the XML document. A good practice is to use the naming rules of your database for the elements in the XML documents.

Non-English letters like éöá are perfectly legal in XML element names, but watch out for problems if your software vendor doesn't support them.

The ":" should not be used in element names because it is reserved to be used for something called namespaces (more later).

### **XML elements can have attributes in the start tag, just like HTML.**

Attributes are used to provide additional information about elements.

## **XML Attributes**

XML elements can have attributes. From HTML you will remember this: <IMG SRC="computer.gif">. The SRC attribute provides additional information about the IMG element.

In HTML (and in XML) attributes provide additional information about elements:

```
  
<a href="demo.asp">
```

Attributes often provide information that is not a part of the data. In the example below, the file type is irrelevant to the data, but important to the software that wants to manipulate the element:

```
<file type="gif">computer.gif</file>
```

### **Quote Styles, "female" or 'female'?**

Attribute values must always be enclosed in quotes, but either single or double quotes can be used. For a person's sex, the person tag can be written like this:

```
<person sex="female">
```

or like this:

```
<person sex='female'>
```

**Note:** If the attribute value itself contains double quotes it is necessary to use single quotes, like in this example:

```
<gangster name='George "Shotgun" Ziegler'>
```

**Note:** If the attribute value itself contains single quotes it is necessary to use double quotes, like in this example:

```
<gangster name="George 'Shotgun' Ziegler">
```

## **Use of Elements vs. Attributes**

Data can be stored in child elements or in attributes. Take a look at these examples:

```
<person sex="female">  
  <firstname>Anna</firstname>  
  <lastname>Smith</lastname>  
</person>
```

```
<person>
```

```

<sex>female</sex>
<firstname>Anna</firstname>
<lastname>Smith</lastname>
</person>

```

In the first example sex is an attribute. In the last, sex is a child element. Both examples provide the same information.

There are no rules about when to use attributes, and when to use child elements. My experience is that attributes are handy in HTML, but in XML you should try to avoid them. Use child elements if the information feels like data.

## Avoid using attributes?

Should you avoid using attributes?

Some of the problems with using attributes are:

- attributes cannot contain multiple values (child elements can)
- attributes are not easily expandable (for future changes)
- attributes cannot describe structures (child elements can)
- attributes are more difficult to manipulate by program code
- attribute values are not easy to test against a Document Type Definition (DTD) - which is used to define the legal elements of an XML document

If you use attributes as containers for data, you end up with documents that are difficult to read and maintain. Try to use **elements** to describe data. Use attributes only to provide information that is not relevant to the data.

Don't end up like this (this is not how XML should be used):

```

<note day="12" month="11" year="2002"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>

```

## An Exception to my Attribute Rule

Rules always have exceptions. My rule about attributes has one exception:

Sometimes I assign ID references to elements. These ID references can be used to access XML elements in much the same way as the NAME or ID attributes in HTML. This example demonstrates this:

```

<messages>
  <note id="p501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>

  <note id="p502">
    <to>Jani</to>
    <from>Tove</from>
    <heading>Re: Reminder</heading>
    <body>I will not!</body>
  </note>
</messages>

```

The ID in these examples is just a counter, or a unique identifier, to identify the different notes in the XML file, and not a part of the note data.

What I am trying to say here is that metadata (data about data) should be stored as attributes, and that data itself should be stored as elements.

### **XML with correct syntax is Well Formed XML.**

XML validated against a DTD is Valid XML.

## **Well Formed XML Documents**

A "Well Formed" XML document has correct XML syntax. "Well Formed" XML document is a document that conforms to the XML syntax rules that were described in the previous chapters:

- XML documents must have a root element
- XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must always be quoted

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

## **Valid XML Documents**

A "Valid" XML document also conforms to a DTD. A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition (DTD):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "InternalNote.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

## **XML DTD**

A DTD defines the legal elements of an XML document. The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements

## **XML Schema**

XML Schema is an XML based alternative to DTD. W3C supports an alternative to DTD called XML Schema.

### **Errors in XML documents will stop your XML program.**

The W3C XML specification states that a program should not continue to process an XML document if it finds an error. The reason is that XML software should be easy to write, and that all XML documents should be compatible.

With HTML it was possible to create documents with lots of errors (like when you forget an end tag). One of the main reasons that HTML browsers are so big and incompatible, is that they have their own ways to figure out what a document should look like when they encounter an HTML error.

With XML this should not be possible.

### Validate your XML - IE Only

To help you validate your xml, we have used Microsoft's XML parser to create an XML validator.

Paste your XML in the text area below, and syntax-check it by pressing the "Validate" button.

```
<?xml version="1.0" ?>

<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

---

### Validate your XML File

You can also validate your XML files by typing the URL of your file into the input field below, and then press the "Validate" button

Filename:

If you want to syntax-check (validate) an error-free XML file, you can paste the following address into the filename field: [http://www.w3schools.com/xml/cd\\_catalog.xml](http://www.w3schools.com/xml/cd_catalog.xml)

**Note:** If you get the error "Access denied" when accessing this file, it is because your Internet Explorer security settings do not allow access across domains!

**Nearly all major browsers have support for XML and XSLT.**

### Viewing XML Files

In Firefox and Internet Explorer:

Open the XML file (typically by clicking on a link) - The XML document will be displayed with color-coded root and child elements. A plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure. To view the raw XML source (without the + and - signs), select "View Page Source" or "View Source" from the browser menu.

**Note: Do not expect XML files to be formatted like HTML documents!**

### Why Does XML Display Like This?

XML documents do not carry information about how to display the data. Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table.

Without any information about how to display the data, most browsers will just display the XML document as it is.

In the next chapters, we will take a look at different solutions to the display problem, using CSS, XSL, JavaScript, and XML Data Islands.

**With CSS (Cascading Style Sheets) you can add display information to an XML document.**

## Displaying your XML Files with CSS?

It is possible to use CSS to format an XML document. Below is an example of how to use a CSS style sheet to format an XML document:

Below is a fraction of the XML file. The second line, `<?xml-stylesheet type="text/css" href="cd_catalog.css"?>`, links the XML file to the CSS file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
  .
  .
  .
</CATALOG>
```

**Note:** Formatting XML with CSS is NOT the future of how to style XML documents. XML document should be styled by using the W3C's XSL standard!

**With XSL you can add display information to your XML document.**

## Displaying XML with XSL

XSL is the preferred style sheet language of XML. XSL (the eXtensible Stylesheet Language) is far more sophisticated than CSS. One way to use XSL is to transform XML into HTML before it is displayed by the browser as demonstrated in these examples:

**View the XML file, the XSL style sheet, and View the result.**

Below is a fraction of the XML file. The second line, `<?xml-stylesheet type="text/xsl" href="simple.xsl"?>`, links the XML file to the XSL file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="simple.xsl"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
      two of our famous Belgian Waffles
    </description>
    <calories>650</calories>
  </food>
</breakfast_menu>
```

**With Internet Explorer, the unofficial `<xml>` tag can be used to create an XML data island.**

XML Data Embedded in HTML . An XML data island is XML data embedded into an HTML page.

Here is how it works; assume we have the following XML document ("note.xml"):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Then, in an HTML document, you can embed the XML file above with the `<xml>` tag. The `id` attribute of the `<xml>` tag defines an ID for the data island, and the `src` attribute points to the XML file to embed:

```
<html>
<body>
<xml id="note" src="note.xml"></xml>
</body>
</html>
```

However, the embedded XML data is, up to this point, not visible for the user.

The next step is to format and display the data in the data island by binding it to HTML elements.

## Bind Data Island to HTML Elements

In the next example, we will embed an XML file called "cd\_catalog.xml" into an HTML file.

The HTML file looks like this:

```
<html>
<body>

<xml id="cdcat" src="cd_catalog.xml"></xml>

<table border="1" datasrc="#cdcat">
<tr>
```

```

<td><span datafld="ARTIST"></span></td>
<td><span datafld="TITLE"></span></td>
</tr>
</table>

</body>
</html>

```

Example explained:

The datasrc attribute of the <table> tag binds the HTML table element to the XML data island. The datasrc attribute refers to the id attribute of the data island.

<td> tags cannot be bound to data, so we are using <span> tags. The <span> tag allows the datafld attribute to refer to the XML element to be displayed. In this case, it is datafld="ARTIST" for the <ARTIST> element and datafld="TITLE" for the <TITLE> element in the XML file. As the XML is read, additional rows are created for each <CD> element.

**To read and update - create and manipulate - an XML document, you will need an XML parser.**

## Microsoft's XML Parser

Microsoft's XML parser is a COM component that comes with Internet Explorer 5 and higher. Once you have installed Internet Explorer, the parser is available to scripts.

Microsoft's XML parser supports all the necessary functions to traverse the node tree, access the nodes and their attribute values, insert and delete nodes, and convert the node tree back to XML.

The following table lists the most commonly used node types supported by Microsoft's XML parser:

Node Type	Example
Processing instruction	<?xml version="1.0"?>
Element	< <b>drink</b> type="beer">Carlsberg</ <b>drink</b> >
Attribute	<b>type</b> ="beer"
Text	Carlsberg

MSXML Parser 2.5 is the XML parser that is shipped with Windows 2000 and IE 5.5.

MSXML Parser 3.0 is the XML parser that is shipped with IE 6.0 and Windows XP.

The MSXML 3.0 parser features:

- JavaScript, VBScript, Perl, VB, Java, C++, etc. support
- Complete XML support
- Full DOM and Namespace support
- DTD and validation
- Complete XSLT and XPath support
- SAX2 support
- Server-safe HTTP

To create an instance of Microsoft's XML parser with JavaScript, use the following code:

```
var xmlDoc=new ActiveXObject("Microsoft.XMLDOM")
```

To create an instance of Microsoft's XML parser with VBScript, use the following code:

```
set xmlDoc=CreateObject("Microsoft.XMLDOM")
```

To create an instance of Microsoft's XML parser in an ASP page (using VBScript), use the following code:

```
set xmlDoc=Server.CreateObject("Microsoft.XMLDOM")
```

The following code loads an existing XML document ("note.xml") into Microsoft's XML parser:

```
<script type="text/javascript">
var xmlDoc=new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load("note.xml")
...
...
...
</script>
```

The first line of the script above creates an instance of the Microsoft XML parser. The third line tells the parser to load an XML document called "note.xml". The second line turns off asynchronous loading, to make sure that the parser will not continue execution of the script before the document is fully loaded.

## XML Parser in Mozilla Browsers

Plain XML documents are displayed in a tree-like structure in Mozilla (just like IE).

Mozilla also supports parsing of XML data using JavaScript. The parsed data can be displayed as HTML.

To create an instance of the XML parser with JavaScript in Mozilla browsers, use the following code:

```
var xmlDoc=document.implementation.createDocument("ns","root",null)
```

The first parameter, *ns*, defines the namespace used for the XML document. The second parameter, *root*, is the XML root element in the XML file. The third parameter, null, is always null because it is not implemented yet.

The following code loads an existing XML document ("note.xml") into Mozillas' XML parser:

```
<script type="text/javascript">
var xmlDoc=document.implementation.createDocument("", "", null);
xmlDoc.load("note.xml");
...
...
...
</script>
```

The first line of the script above creates an instance of the XML parser. The second line tells the parser to load an XML document called "note.xml".

## Loading an XML File - A Cross browser Example

The following example is a cross browser example that loads an existing XML document ("note.xml") into the XML parser:

```
<html>
<head>
<script type="text/javascript">
var xmlDoc
function loadXML()
{
//load xml file
// code for IE
```

```

if (window.ActiveXObject)
{
    xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
    xmlDoc.async=false;
    xmlDoc.load("note.xml");
    getmessage()
}
// code for Mozilla, etc.
else if (document.implementation &&
document.implementation.createDocument)
{
    xmlDoc= document.implementation.createDocument("", "", null);
    xmlDoc.load("note.xml");
    xmlDoc.onload=getmessage
}
else
{
    alert('Your browser cannot handle this script');
}
}

function getmessage()
{
document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].firstChild.nodeValue
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].firstChild.nodeValue
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].firstChild.nodeValue
}
</script>
</head>
<body onload="loadXML()" bgcolor="yellow">
<h1>W3Schools Internal Note</h1>
<p><b>To:</b> <span id="to"></span><br />
<b>From:</b> <span id="from"></span>
<hr />
<b>Message:</b> <span id="message"></span>
</p>
</body>
</html>

```

## Loading XML Text Into the Parser

Internet Explorer supports two ways of loading XML into a document object: the load() method and the loadXML() method. The load() method loads an XML file and the loadXML() method loads a text string that contains XML code.

The following code loads a text string into Microsoft's XML parser:

```

<script type="text/javascript">
var txt=<note>
txt=txt+<to>Tove</to><from>Jani</from>
txt=txt+<heading>Reminder</heading>
txt=txt+<body>Don't forget me this weekend!</body>
txt=txt+</note>
var xmlDoc=new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.loadXML(txt)
...
...
...
</script>

```

## **Some real-life examples of how XML can be used to carry information.**

XMLNews is a specification for exchanging news and other information. Using such a standard makes it easier for both news producers and news consumers to produce, receive, and archive any kind of news information across different hardware, software, and programming languages.

### **An example XML News document:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<nitf>

<head>
<title>Colombia Earthquake</title>
</head>

<body>

<headline>
    <h1>143 Dead in Colombia Earthquake</h1>
</headline>
<byline>
    <bytag>By Jared Kotler, Associated Press Writer</bytag>
</byline>
<dateline>
    <location>Bogota, Colombia</location>
    <date>Monday January 25 1999 7:28 ET</date>
</dateline>

</body>

</nitf>
```

**XML Namespaces provide a method to avoid element name conflicts.**

### **Name Conflicts**

Since element names in XML are not predefined, a name conflict will occur when two different documents use the same element names.

This XML document carries information in a table:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

This XML document carries information about a table (a piece of furniture):

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

If these two XML documents were added together, there would be an element name conflict because both documents contain a `<table>` element with different content and definition.

### **Solving Name Conflicts Using a Prefix**

This XML document carries information in a table:

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

This XML document carries information about a piece of furniture:

```
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Now there will be no name conflict because the two documents use a different name for their `<table>` element (`<h:table>` and `<f:table>`).

By using a prefix, we have created two different types of `<table>` elements.

## Using Namespaces

This XML document carries information in a table:

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

This XML document carries information about a piece of furniture:

```
<f:table xmlns:f="http://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Instead of using only prefixes, we have added an `xmlns` attribute to the `<table>` tag to give the prefix a qualified name associated with a namespace.

## The XML Namespace (`xmlns`) Attribute

The XML namespace attribute is placed in the start tag of an element and has the following syntax:

```
xmlns:namespace-prefix="namespaceURI"
```

When a namespace is defined in the start tag of an element, all child elements with the same prefix are associated with the same namespace.

Note that the address used to identify the namespace is not used by the parser to look up information. The only purpose is to give the namespace a unique name. However, very often companies use the namespace as a pointer to a real Web page containing information about the namespace.  
Uniform Resource Identifier (URI)

A **Uniform Resource Identifier** (URI) is a string of characters which identifies an Internet Resource. The most common URI is the **Uniform Resource Locator** (URL) which identifies an Internet domain address. Another, not so common type of URI is the **Universal Resource Name** (URN). In our examples we will only use URLs.

## Default Namespaces

Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:

```
xmlns="namespaceURI"
```

This XML document carries information in a table:

```
<table xmlns="http://www.w3.org/TR/html4/">
<tr>
<td>Apples</td>
<td>Bananas</td>
</tr>
</table>
```

This XML document carries information about a piece of furniture:

```
<table xmlns="http://www.w3schools.com/furniture">
<name>African Coffee Table</name>
<width>80</width>
<length>120</length>
</table>
```

## Namespaces in Real Use

When you start using XSL, you will soon see namespaces in real use. XSL style sheets are used to transform XML documents into other formats, like HTML.

If you take a close look at the XSL document below, you will see that most of the tags are HTML tags. The tags that are not HTML tags have the prefix xsl, identified by the namespace "http://www.w3.org/1999/XSL/Transform":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr>
<th align="left">Title</th>
<th align="left">Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
```

```
</xsl:stylesheet>
```

**All text in an XML document will be parsed by the parser.**

Only text inside a CDATA section will be ignored by the parser.

## Parsed Data

**XML parsers normally parse all the text in an XML document.**

When an XML element is parsed, the text between the XML tags is also parsed:

```
<message>This text is also parsed</message>
```

The parser does this because XML elements can contain other elements, as in this example, where the `<name>` element contains two other elements (first and last):

```
<name><first>Bill</first><last>Gates</last></name>
```

and the parser will break it up into sub-elements like this:

```
<name>
  <first>Bill</first>
  <last>Gates</last>
</name>
```

## Escape Characters

**Illegal XML characters have to be replaced by entity references.**

If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element. You cannot write something like this:

```
<message>if salary < 1000 then</message>
```

To avoid this, you have to replace the "<" character with an entity reference, like this:

```
<message>if salary &lt; 1000 then</message>
```

There are 5 predefined entity references in XML:

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

**Note:** Only the characters "<" and "&" are strictly illegal in XML. Apostrophes, quotation marks and greater than signs are legal, but it is a good habit to replace them.

## CDATA

**Everything inside a CDATA section is ignored by the parser.**

If your text contains a lot of "<" or "&" characters - as program code often does - the XML element can be defined as a CDATA section.

A CDATA section starts with "<![CDATA[" and ends with "]]>":

```
<script>
<![CDATA[
function matchwo(a,b)
{
if (a < b && a < 0) then
{
    return 1
}
else
{
    return 0
}
]]>
</script>
```

In the example above, everything inside the CDATA section is ignored by the parser.

#### **Notes on CDATA sections:**

A CDATA section cannot contain the string "]]>", therefore, nested CDATA sections are not allowed.

Also make sure there are no spaces or line breaks inside the "]]>" string.

**XML documents may contain foreign characters, like Norwegian æ ø å , or French è è é.**

**To let your XML parser understand these characters, you should save your XML documents as Unicode.**

#### **Windows 2000 Notepad**

Windows 2000 Notepad can save files as Unicode.

Save the XML file below as Unicode (note that the document does not contain any encoding attribute):

```
<?xml version="1.0"?>
<note>
  <from>Jani</from>
  <to>Tove</to>
  <message>Norwegian: æøå. French: èèé</message>
</note>
```

The file above, note\_encode\_none\_u.xml will NOT generate an error in IE 5+, Firefox, or Opera, but it WILL generate an error in Netscape 6.2.

#### **Conclusion**

The conclusion is that the encoding attribute has to specify the encoding used when the document was saved. My best advice to avoid errors is:

- Use an editor that supports encoding
- Make sure you know what encoding it uses
- Use the same encoding attribute in your XML documents

**XML can be generated on a server without installing any XML controls.**

## Storing XML on the Server

XML files can be stored on an Internet server exactly the same way as HTML files.

Start Windows Notepad and write the following lines:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <from>Jani</from>
  <to>Tove</to>
  <message>Remember me this weekend</message>
</note>
```

Save the file on your web server with a proper name like "note.xml".

## Generating XML with ASP

**XML can be generated on a server without any installed XML software.**

To generate an XML response from the server - simply write the following code and save it as an ASP file on the web server:

```
<%
response.ContentType="text/xml"
response.Write("<?xml version='1.0' encoding='ISO-8859-1'?>")
response.Write("<note>")
response.Write("<from>Jani</from>")
response.Write("<to>Tove</to>")
response.Write("<message>Remember me this weekend</message>")
response.Write("</note>")
%>
```

Note that the content type of the response must be set to "text/xml".

## Getting XML From a Database

**XML can be generated from a database without any installed XML software.**

To generate an XML database response from the server, simply write the following code and save it as an ASP file on the web server:

```
<%
response.ContentType = "text/xml"
set conn=Server.CreateObject("ADODB.Connection")
conn.provider="Microsoft.Jet.OLEDB.4.0;"
conn.open server.mappath("/db/database.mdb")
sql="select fname, lname from tblGuestBook"
set rs=Conn.Execute(sql)
rs.MoveFirst()
response.write("<?xml version='1.0' encoding='ISO-8859-1'?>")
response.write("<guestbook>")
while (not rs.EOF)
  response.write("<guest>")
  response.write("<fname>" & rs("fname") & "</fname>")
  response.write("<lname>" & rs("lname") & "</lname>")
  response.write("</guest>")
  rs.MoveNext()
wend
```

```

rs.close()
conn.close()
response.write("</guestbook>")
%>

```

This chapter demonstrates a small framework for an XML application.

Note: This example uses a Data Island, which only works in Internet Explorer.

## The XML Example Document

Look at the following XML document ("cd\_catalog.xml"), that represents a CD catalog:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  .
  .
  ... more ...
  .

```

## Load the XML Document Into a Data Island

**A Data Island can be used to access the XML file.**

To get your XML document "inside" an HTML page, add an XML Data Island to the HTML page:

```

<xml src="cd_catalog.xml" id="xmldso" async="false">
</xml>

```

With the example code above, the XML file "cd\_catalog.xml" will be loaded into an "invisible" Data Island called "xmldso". The `async="false"` attribute is added to make sure that all the XML data is loaded before any other HTML processing takes place.

## Bind the Data Island to an HTML Table

To make the XML data visible on the HTML page, you must "bind" the Data Island to an HTML element.

To bind the XML data to an HTML table, add a `datasrc` attribute to the `table` element, and add `datafld` attributes to the `span` elements inside the table data:

```


| <span datafld="TITLE"></span> | <span datafld="ARTIST"></span> | <span datafld="YEAR"></span> |
|-------------------------------|--------------------------------|------------------------------|
|-------------------------------|--------------------------------|------------------------------|


```

## Bind the Data Island to <span> or <div> Elements

<span> or <div> elements can be used to display XML data.

You don't have to use the HTML table element to display XML data. Data from a Data Island can be displayed anywhere on an HTML page.

All you have to do is to add some <span> or <div> elements to your page. Use the datasrc attribute to bind the elements to the Data Island, and the datafld attribute to bind each element to an XML element, like this:

```
<br />Title:  
<span datasrc="#xmldso" datafld="TITLE"></span>  
<br />Artist:  
<span datasrc="#xmldso" datafld="ARTIST"></span>  
<br />Year:  
<span datasrc="#xmldso" datafld="YEAR"></span>
```

or like this:

```
<br />Title:  
<div datasrc="#xmldso" datafld="TITLE"></div>  
<br />Artist:  
<div datasrc="#xmldso" datafld="ARTIST"></div>  
<br />Year:  
<div datasrc="#xmldso" datafld="YEAR"></div>
```

Note that if you use an HTML <div> element, the data will be displayed on a new line.

With the examples above, you will only see one line of your XML data. To navigate to the next line of data, you have to add some scripting to your code.

## Add a Navigation Script

**Navigation has to be performed by a script.**

To add navigation to the XML Data Island, create a script that calls the movenext() and moveprevious() methods of the Data Island.

```
<script type="text/javascript">  
function movenext()  
{  
x=xmldso.recordset  
if (x.absolutePosition < x.recordCount)  
{  
    x.movenext()  
}  
}  
function moveprevious()  
{  
x=xmldso.recordset  
if (x.absolutePosition > 1)  
{  
    x.moveprevious()  
}  
}  
</script>
```

## All Together Now

## **With a little creativity you can create a full application.**

If you use what you have learned on this page, and a little imagination, you can easily develop this into a full application.

### **What is an HTTP Request?**

With an HTTP request, a web page can make a request to, and get a response from a web server - without reloading the page. The user will stay on the same page, and he or she will not notice that scripts might request pages, or send data to a server in the background.

**By using the XMLHttpRequest object, a web developer can change a page with data from the server after the page has loaded.**

Google Suggest is using the XMLHttpRequest object to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a **list of suggestions**.

### **Creating an XMLHttpRequest Object**

For Mozilla, Firefox, Safari, Opera, and Netscape:

```
var xmlhttp=new XMLHttpRequest()
```

For Internet Explorer:

```
var xmlhttp=new ActiveXObject("Microsoft.XMLHTTP")
```

### **Example**

```
<script type="text/javascript">
var xmlhttp
function loadXMLDoc(url)
{
xmlhttp=null
// code for Mozilla, etc.
if (window.XMLHttpRequest)
{
  xmlhttp=new XMLHttpRequest()
}
// code for IE
else if (window.ActiveXObject)
{
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP")
}
if (xmlhttp!=null)
{
  xmlhttp.onreadystatechange=state_Change
  xmlhttp.open("GET",url,true)
  xmlhttp.send(null)
}
else
{
  alert("Your browser does not support XMLHTTP.")
}
}
function state_Change()
{
// if xmlhttp shows "loaded"
if (xmlhttp.readyState==4)
```

```

{
// if "OK"
if (xmlhttp.status==200)
{
// ...some code here...
}
else
{
alert("Problem retrieving XML data")
}
}
</script>

```

Note: An important property in the example above is the onreadystatechange property. This property is an event handler which is triggered each time the state of the request changes. The states run from 0 (uninitialized) to 4 (complete). By having the function xmlhttpChange() check for the state changing, we can tell when the process is complete and continue only if it has been successful.

## Why are we Using Async in our Examples?

All the examples here use the async mode (the third parameter of open() set to true).

The async parameter specifies whether the request should be handled asynchronously or not. True means that script continues to run after the send() method, without waiting for a response from the server. false means that the script waits for a response before continuing script processing. By setting this parameter to false, you run the risk of having your script hang if there is a network or server problem, or if the request is long (the UI locks while the request is being made) a user may even see the "Not Responding" message. It is safer to send asynchronously and design your code around the onreadystatechange event!

## The XMLHttpRequest Object Reference

### Methods

Method	Description
abort()	Cancels the current request
getAllResponseHeaders()	Returns the complete set of http headers as a string
getResponseHeader("headername")	Returns the value of the specified http header
open("method","URL",async,"uname","pswd")	<p>Specifies the method, URL, and other optional attributes of a request</p> <p>The method parameter can have a value of "GET", "POST", or "PUT" (use "GET" when requesting data and use "POST" when sending data (especially if the length of the data is greater than 512 bytes).</p> <p>The URL parameter may be either a relative or complete URL.</p> <p>The async parameter specifies whether the request should be handled asynchronously or not. true means that script processing carries on after the send() method, without waiting for a response. false means that the script waits for a response before continuing script processing</p>
send(content)	Sends the request
setRequestHeader("label","value")	Adds a label/value pair to the http header to be sent

## Properties

Property	Description
onreadystatechange	An event handler for an event that fires at every state change
readyState	Returns the state of the object:  0 = uninitialized 1 = loading 2 = loaded 3 = interactive 4 = complete
responseText	Returns the response as a string
responseXML	Returns the response as XML. This property returns an XML document object, which can be examined and parsed using W3C DOM node tree methods and properties
status	Returns the status as a number (e.g. 404 for "Not Found" or 200 for "OK")
statusText	Returns the status as a string (e.g. "Not Found" or "OK")

**Usually, we save data in databases. However, if we want to make the data more portable, we can store the data in an XML file.**

## Create and Save an XML File

Storing data in XML files is useful if the data is to be sent to applications on non-Windows platforms. Remember that XML is portable across all platforms and the data will not need to be converted!

First we will learn how to create and save an XML file. The XML file below will be named "test.xml" and will be stored in the c directory on the server. We will use ASP and Microsoft's XMLElement object to create and save the XML file:

```
<%
Dim xmlDoc, rootEl, child1, child2, p
'Create an XML document
Set xmlDoc = Server.CreateObject("Microsoft.XMLDOM")
'Create a root element and append it to the document
Set rootEl = xmlDoc.createElement("root")
xmlDoc.appendChild rootEl
'Create and append child elements
Set child1 = xmlDoc.createElement("child1")
Set child2 = xmlDoc.createElement("child2")
rootEl.appendChild child1
rootEl.appendChild child2
'Add an XML processing instruction
'and insert it before the root element
Set p=xmlDoc.createProcessingInstruction("xml","version='1.0'")
xmlDoc.insertBefore p,xmlDoc.childNodes(0)
'Save the XML file to the c directory
xmlDoc.Save "c:\test.xml"
%>
```

If you open the saved XML file it will look something like this ("test.xml"):

```
<?xml version="1.0"?>
<root>
  <child1 />
```

```
<child2 />  
</root>
```

## Real Form Example

Now, we will look at a real HTML form example.

We will first look at the HTML form that will be used in this example: The HTML form below asks for the user's name, country, and e-mail address. This information will then be written to an XML file for storage.

"customers.htm":

```
<html>  
<body>  
<form action="saveForm.asp" method="post">  
<p><b>Enter your contact information</b></p>  
First Name: <input type="text" id="fname" name="fname"><br />  
Last Name: <input type="text" id="lname" name="lname"><br />  
Country: <input type="text" id="country" name="country"><br />  
Email: <input type="text" id="email" name="email"><br />  
<input type="submit" id="btn_sub" name="btn_sub" value="Submit">  
<input type="reset" id="btn_res" name="btn_res" value="Reset">  
</form>  
</body>  
</html>
```

The action for the HTML form above is set to "saveForm.asp". The "saveForm.asp" file is an ASP page that will loop through the form fields and store their values in an XML file:

```
<%  
dim xmlDoc  
dim rootEl,fieldName,fieldValue,attID  
dim p,i  
'Do not stop if an error occurs  
On Error Resume Next  
Set xmlDoc = server.CreateObject("Microsoft.XMLDOM")  
xmlDoc.preserveWhiteSpace=true  
'Create a root element and append it to the document  
Set rootEl = xmlDoc.createElement("customer")  
xmlDoc.appendChild rootEl  
'Loop through the form collection  
for i = 1 To Request.Form.Count  
    'Eliminate button elements in the form  
    if instr(1,Request.Form.Key(i),"btn_")=0 then  
        'Create a field and a value element, and an id attribute  
        Set fieldName = xmlDoc.createElement("field")  
        Set fieldValue = xmlDoc.createElement("value")  
        Set attID = xmlDoc.createAttribute("id")  
        'Set the value of the id attribute equal to the name of  
        'the current form field  
        attID.Text = Request.Form.Key(i)  
        'Append the id attribute to the field element  
        fieldName.setAttributeNode attID  
        'Set the value of the value element equal to  
        'the value of the current form field  
        fieldValue.Text = Request.Form(i)  
        'Append the field element as a child of the root element  
        rootEl.appendChild fieldName  
        'Append the value element as a child of the field element  
        fieldName.appendChild fieldValue
```

```

    end if
next
'Add an XML processing instruction
'and insert it before the root element
Set p = xmlDoc.createProcessingInstruction("xml", "version='1.0'")
xmlDoc.insertBefore p,xmlDoc.childNodes(0)
'Save the XML file
xmlDoc.save "c:\Customer.xml"
'Release all object references
set xmlDoc=nothing
set rootEl=nothing
set fieldName=nothing
set fieldValue=nothing
set attID=nothing
set p=nothing
'Test to see if an error occurred
if err.number<>0 then
    response.write("Error: No information saved.")
else
    response.write("Your information has been saved.")
end if
%>

```

**Note:** If the XML file name specified already exists, it will be overwritten!

The XML file that will be produced by the code above will look something like this ("Customer.xml"):

```

<?xml version="1.0" ?>
<customer>
    <field id="firstName">
        <value>Hege</value>
    </field>
    <field id="lastName">
        <value>Rfsnes</value>
    </field>
    <field id="country">
        <value>Norway</value>
    </field>
    <field id="email">
        <value>mymail@myaddress.com</value>
    </field>
</customer>

```

**Internet Explorer 5 introduced DHTML behaviors. Behaviors are a way to add DHTML functionality to HTML elements with the ease of CSS.**

### Behaviors - What are They?

IE5 introduced DHTML behaviors. Behaviors are a way to add DHTML functionality to HTML elements with the ease of CSS.

How do behaviors work? By using XML we can link behaviors to any element in a web page and manipulate that element.

DHTML behaviors do not use a `<script>` tag. Instead, they are using a CSS attribute called "behavior". This "behavior" specifies a URL to an HTC file which contains the actual behavior (The HTC file is written in XML).

### Syntax

```
behavior: url(some_filename.htc)
```

Note: The behavior attribute is only supported by IE 5 and higher, all other browsers will ignore it. This means that Mozilla, Firefox, Netscape and other browsers will only see the regular content and IE 5+ can see the DHTML behaviors.

## Example

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 the event handlers for the script.

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(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.htm" 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>
```

### **XML Summary**

This tutorial has taught you how to use XML to describe data.

You have learned that XML should be used to separate the data from the HTML code.

You have also learned that XML can be used to exchange, share, and store data.

# JavaScript

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Mozilla, Firefox, Netscape, Opera. It is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more. Basic understanding of HTML / XHTML will help to study Javascript .

## 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)
- A JavaScript consists of lines of executable computer code
- A 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 will save 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

## JavaScript How To ...

The HTML `<script>` tag is used to insert a JavaScript into an HTML 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 (also 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 type="text/javascript">` 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.

### Ending Statements With a Semicolon?

With traditional programming languages, like C++ and Java, each code statement has to end with a semicolon.

Many programmers continue this habit when writing JavaScript, but in general, semicolons are **optional!** However, semicolons are required if you want to put more than one statement on a single line.

### How to Handle Older Browsers

Browsers that do not support JavaScript will display the script as page content. To prevent them from doing this, we may use the HTML comment tag:

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

The two forward slashes at the end of comment line (//) are a JavaScript comment symbol. This prevents the JavaScript compiler from compiling the line.

## 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.

## 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 Variables

A variable is a "container" for information you want to store.

### Examples

Variable

Variables are used to store data. This example will show you how.

### Variables

A variable is a "container" for information you want to store. A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.

Rules for variable names:

- Variable names are case sensitive
- They must begin with a letter or the underscore character

**IMPORTANT!** JavaScript is case-sensitive! A variable named strname is not the same as a variable named STRNAME!

### Declare a Variable

You can create a variable with the var statement:

```
var strname = some value
```

You can also create a variable without the var statement:

```
strname = some value
```

### Assign a Value to a Variable

You can assign a value to a variable like this:

```
var strname = "Hege"
```

Or like this:

```
strname = "Hege"
```

The variable name is on the left side of the expression and the value you want to assign to the variable is on the right. Now the variable "strname" has the value "Hege".

## Lifetime of 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 If...Else Statements

Conditional statements in JavaScript are used to perform different actions based on different conditions.

## 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("Good morning")
}
else if (time>10 && time<16)
{
document.write("Good day")
}
else
{
document.write("Hello World!")
}
</script>
```

## JavaScript Switch Statement

Conditional statements in JavaScript are used to perform different actions based on different conditions.

### 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 Operators

### Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6

--	Decrement	x=5 x--	x=4
----	-----------	------------	-----

## Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

## Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
====	is equal to (checks for both value and type)	x=5 y="5"  x==y returns true x====y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

## Logical Operators

Operator	Description	Example
&&	and	x=6 y=3  (x < 10 && y > 1) returns true
	or	x=6 y=3  (x==5    y==5) returns false
!	not	x=6 y=3  !(x==y) returns true

## String Operator

A string is most often text, for example "Hello World!". To stick two or more string variables together, use the + operator.

```
txt1="What a very"
txt2="nice day!"
txt3=txt1+txt2
```

The variable txt3 now contains "What a very nice day!".

To add a space between two string variables, insert a space into the expression, OR in one of the strings.

```
txt1="What a very"
txt2="nice day!"
```

```
txt3=txt1+" "+txt2  
or  
txt1="What a very "  
txt2="nice day!"  
txt3=txt1+txt2
```

The variable txt3 now contains "What a very nice day!".

## 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 is equal to PRES, then put the string "Dear President " in the variable named greeting. If the variable visitor is not equal to PRES, then put the string "Dear " into the variable named greeting.

## JavaScript Popup Boxes

In JavaScript we can create three kinds of popup boxes: Alert box, Confirm box, and Prompt box.

### 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.

### JavaScript Functions

To keep the browser from executing a script as soon as the page is loaded, you can write your script as a function.

A function contains some code that will be executed only 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 are defined at the beginning of a page, 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 written 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.

# 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.

## 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.

### 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. 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.

### 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

The following table lists the events recognized by JavaScript:

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">  
  
</a>
```

## **JavaScript Try...Catch Statement**

The try...catch statement allows you to test a block of code for errors.

## 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.

### 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.

### 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()
{
addalert("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 \& me are singing!")
```

The example above will produce the following output:

```
You & me 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!" )
```

## Comments

You can add comments to your script by using two slashes //:

```
//this is a comment  
document.write("Hello World!")
```

or by using /\* and \*/ (this creates a multi-line comment block):

```
/* This is a comment  
block. It contains  
several lines */  
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.

### 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!
```

### 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!

## JavaScript Date Object

The Date object is used to work with dates and times.

### Defining Dates

The Date object is used to work with dates and times.

We define a Date object with the new keyword. The following code line defines 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!

## Manipulate 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!

## Comparing 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")
```

## 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!

## JavaScript Array Object

The Array object is used to store a set of values in a single variable name.

### Defining Arrays

The Array object is used to store a set of values in a single variable name.

We define an Array object with the new keyword. The following code line defines an Array object called myArray:

```
var myArray=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).

```
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"
```

```
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.

## Accessing Arrays

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 Existing Arrays

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
```

## 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!

## JavaScript Boolean Object

The Boolean object is used to convert a non-Boolean value to a Boolean value (true or false).

### Boolean Object

The Boolean object is an object wrapper for a Boolean value.

The Boolean object is used to convert a non-Boolean value to a Boolean value (true or false).

We define a Boolean object with the new keyword. The following code line defines 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")
```

### 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!

## JavaScript Math Object

The Math object allows you to perform common mathematical tasks.

### Math Object

The Math object allows you to perform common mathematical tasks.

The Math object includes several mathematical values and functions. You do not need to define the Math object before using it.

## **Mathematical Values**

JavaScript provides eight mathematical values (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 values 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 values that can be accessed from the Math object there are also several functions (methods) available.

### **Examples of functions (methods):**

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:

```
0.408020536244193
```

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:

```
9
```

## 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!

## 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
<a href="#">Window</a>	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
<a href="#">Navigator</a>	Contains information about the client's browser
<a href="#">Screen</a>	Contains information about the client's display screen
<a href="#">History</a>	Contains the visited URLs in the browser window
<a href="#">Location</a>	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
<a href="#">Document</a>	Represents the entire HTML document and can be used to access all elements in a page
<a href="#">Anchor</a>	Represents an <a> element
<a href="#">Area</a>	Represents an <area> element inside an image-map
<a href="#">Base</a>	Represents a <base> element
<a href="#">Body</a>	Represents the <body> element
<a href="#">Button</a>	Represents a <button> element
<a href="#">Event</a>	Represents the state of an event
<a href="#">Form</a>	Represents a <form> element
<a href="#">Frame</a>	Represents a <frame> element
<a href="#">Frameset</a>	Represents a <frameset> element

<u>Iframe</u>	Represents an <iframe> element
<u>Image</u>	Represents an <img> element
<u>Input button</u>	Represents a button in an HTML form
<u>Input checkbox</u>	Represents a checkbox in an HTML form
<u>Input file</u>	Represents a fileupload in an HTML form
<u>Input hidden</u>	Represents a hidden field in an HTML form
<u>Input password</u>	Represents a password field in an HTML form
<u>Input radio</u>	Represents a radio button in an HTML form
<u>Input reset</u>	Represents a reset button in an HTML form
<u>Input submit</u>	Represents a submit button in an HTML form
<u>Input text</u>	Represents a text-input field in an HTML form
<u>Link</u>	Represents a <link> element
<u>Meta</u>	Represents a <meta> element
<u>Option</u>	Represents an <option> element
<u>Select</u>	Represents a selection list in an HTML form
<u>Style</u>	Represents an individual style statement
<u>Table</u>	Represents a <table> element
<u>TableData</u>	Represents a <td> element
<u>TableRow</u>	Represents a <tr> element
<u>Textarea</u>	Represents a <textarea> element

## JavaScript Browser Detection

The JavaScript Navigator object contains information about the visitor's browser.

### 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 - specially 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.

## 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)
}
```

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 null
}

```

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 null.

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)
    {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 null
}
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)
}
function checkCookie()
{

```

```

username=getCookie('username')
if (username!=null)
  {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.

### 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">

</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!">
<img border="0" alt="Visit W3Schools!">
src="b_pink.gif" name="b1"
onmouseover="mouseOver()"
onmouseout="mouseOut()" />
</a>
</body>
</html>
```

## JavaScript Image Maps

### 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

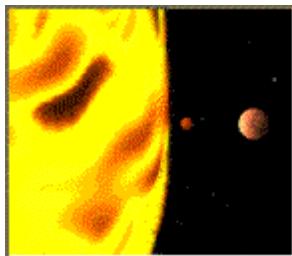
The example below demonstrates how to create an HTML image map, with clickable regions. Each of the regions is a hyperlink:

```

<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`, `onDbClick`, `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>


<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.

### 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.

### 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 in 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")`.

## 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.

# JAVA

## Introduction

The Java programming language and environment is designed to solve a number of problems in modern programming practice. Java started as a part of a larger project to develop advanced software for consumer electronics. These devices are small, reliable, portable, distributed, real-time embedded systems. When we started the project we intended to use C++, but encountered a number of problems. Initially these were just compiler technology problems, but as time passed more problems emerged that were best solved by changing the language.

## What is Java ?

**Java** is a simple, object-oriented, network-savvy, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language.

One way to characterize a system is with a set of buzzwords. We use a standard set of them in describing Java. Here's an explanation of what we mean by those buzzwords and the problems we were trying to solve.

Archimedes Inc. is a fictitious software company that produces software to teach about basic physics. This software is designed to interact with the user, providing not only text and illustrations in the manner of a traditional textbook, but also a set of software lab benches on which experiments can be set up and their behavior simulated. The most basic experiment allows students to put together levers and pulleys and see how they act. The italicized narrative of the trials and tribulations of the Archimedes' designers is used here to provide examples of Java language concepts.

## Simple

We wanted to build a system that could be programmed easily without a lot of esoteric training and which leveraged today's standard practice. Most programmers working these days use C, and most programmers doing object-oriented programming use C++. So even though we found that C++ was unsuitable, we designed Java as closely to C++ as possible in order to make the system more comprehensible.

Java omits many rarely used, poorly understood, confusing features of C++ that in our experience bring more grief than benefit. These omitted features primarily consist of operator overloading (although the Java language does have method overloading), multiple inheritance, and extensive automatic coercions.

We added automatic garbage collection, thereby simplifying the task of Java programming but making the system somewhat more complicated. A common source of complexity in many C and C++ applications is storage management: the allocation and freeing of memory. By virtue of having automatic garbage collection (periodic freeing of memory not being referenced) the Java language not only makes the programming task easier, it also dramatically cuts down on bugs.

The folks at Archimedes wanted to spend their time thinking about levers and pulleys, but instead spent a lot of time on mundane programming tasks. Their central expertise was teaching, not programming. One of the most complicated of these programming tasks was figuring out where memory was being wasted across their 20K lines of code.

Another aspect of being simple is being small. One of the goals of Java is to enable the construction of software that can run stand-alone in small machines. The Java interpreter and standard libraries have a small footprint. A small size is important for use in embedded systems and so Java can be easily downloaded over the net.

## **Object-Oriented**

This is, unfortunately, one of the most overused buzzwords in the industry. But object-oriented design is very powerful because it facilitates the clean definition of interfaces and makes it possible to provide reusable "software ICs."

Simply stated, object-oriented design is a technique that focuses design on the data (=objects) and on the interfaces to it. To make an analogy with carpentry, an "object-oriented" carpenter would be mostly concerned with the chair he was building, and secondarily with the tools used to make it; a "non-object-oriented" carpenter would think primarily of his tools. Object-oriented design is also the mechanism for defining how modules "plug and play."

The object-oriented facilities of Java are essentially those of C++, with extensions from Objective C for more dynamic method resolution.

The folks at Archimedes had lots of things in their simulation, among them ropes and elastic bands. In their initial C version of the product, they ended up with a pretty big system because they had to write separate software for describing ropes versus elastic bands. When they rewrote their application in an object-oriented style, they found they could define one basic object that represented the common aspects of ropes and elastic bands, and then ropes and elastic bands were defined as variations (subclasses) of the basic type. When it came time to add chains, it was a snap because they could build on what had been written before, rather than writing a whole new object simulation.

## **Network-Savvy**

Java has an extensive library of routines for coping easily with TCP/IP protocols like HTTP and FTP. This makes creating network connections much easier than in C or C++. Java applications can open and access objects across the net via URLs with the same ease that programmers are used to when accessing a local file system.

The folks at Archimedes initially built their stuff for CD ROM. But they had some ideas for interactive learning games that they wanted to try out for their next product. For example, they wanted to allow students on different computers to cooperate in building a machine to be simulated. But all the networking systems they'd seen were complicated and required esoteric software specialists. So they gave up.

## **Robust**

Java is intended for writing programs that must be reliable in a variety of ways. Java puts a lot of emphasis on early checking for possible problems, later dynamic (runtime) checking, and eliminating situations that are error prone.

One of the advantages of a strongly typed language (like C++) is that it allows extensive compile-time checking so bugs can be found early. Unfortunately, C++ inherits a number of loopholes in compile-time checking from C, which is relatively lax (particularly method/procedure declarations). In Java, we require declarations and do not support C-style implicit declarations.

The linker understands the type system and repeats many of the type checks done by the compiler to guard against version mismatch problems.

The single biggest difference between Java and C/C++ is that Java has a pointer model that eliminates the possibility of overwriting memory and corrupting data. Instead of pointer arithmetic, Java has true arrays. This allows subscript checking to be performed. In addition, it is not possible to turn an arbitrary integer into a pointer by casting.

The folks at Archimedes had their application basically working in C pretty quickly. But their schedule kept slipping because of all the small bugs that kept slipping through. They had lots of trouble with memory corruption, versions out-of-sync and interface mismatches. What they gained

because C let them pull strange tricks in their code, they paid for in quality assurance time. They also had to reissue their software after the first release because of all the bugs that slipped through.

While Java doesn't make the QA problem go away, it does make it significantly easier.

Very dynamic languages like Lisp, TCL and Smalltalk are often used for prototyping. One of the reasons for their success at this is that they are very robust: you don't have to worry about freeing or corrupting memory. Java programmers can be relatively fearless about dealing with memory because they don't have to worry about it getting corrupted. Because there are no pointers in Java, programs can't accidentally overwrite the end of a memory buffer. Java programs also cannot gain unauthorized access to memory, which could happen in C or C++.

One reason that dynamic languages are good for prototyping is that they don't require you to pin down decisions too early. Java uses another approach to solve this dilemma; Java forces you to make choices explicitly because it has static typing, which the compiler enforces. Along with these choices comes a lot of assistance: you can write method invocations and if you get something wrong, you are informed about it at compile time. You don't have to worry about method invocation error.

## **Secure**

Java is intended for use in networked/distributed environments. Toward that end, a lot of emphasis has been placed on security. Java enables the construction of virus-free, tamper-free systems. The authentication techniques are based on public-key encryption.

There is a strong interplay between "robust" and "secure." For example, the changes to the semantics of pointers make it impossible for applications to forge access to data structures or to access private data in objects that they do not have access to. This closes the door on most activities of viruses.

Someone wrote an interesting "patch" to the PC version of the Archimedes system. They posted this patch to one of the major bulletin boards. Since it was easily available and added some interesting features to the system, lots of people downloaded it. The folks at Archimedes hadn't checked it out, but it seemed to work. Until the next April 1st, when thousands of folks discovered rude pictures popping up in their children's lessons. Needless to say, even though they were in no way responsible for the incident, the folks at Archimedes still had a lot of damage to control.

## **Architecture Neutral**

Java was designed to support applications on networks. In general, networks are composed of a variety of systems with a variety of CPU and operating system architectures. To enable a Java application to execute anywhere on the network, the compiler generates an architecture-neutral object file format--the compiled code is executable on many processors, given the presence of the Java runtime system.

This is useful not only for networks but also for single system software distribution. In the present personal computer market, application writers have to produce versions of their application that are compatible with the IBM PC and with the Apple Macintosh. With the PC market (through Windows/NT) diversifying into many CPU architectures, and Apple moving off the 680x0 toward the PowerPC, production of software that runs on all platforms becomes nearly impossible. With Java, the same version of the application runs on all platforms.

The Java compiler does this by generating bytecode instructions which have nothing to do with a particular computer architecture. Rather, they are designed to be both easy to interpret on any machine and easily translated into native machine code on the fly.

Archimedes is a small company. They started out producing their software for the PC since that was the largest market. After a while, they were a large enough company that they could afford to do a port to the Macintosh, but it was a pretty big effort and didn't really pay off. They couldn't

afford to port to the PowerPC Macintosh or MIPS NT machine. They couldn't "catch the new wave" as it was happening, and a competitor jumped in...

## Portable

Being architecture neutral is a big chunk of being portable, but there's more to it than that. Unlike C and C++, there are no "implementation dependent" aspects of the specification. The sizes of the primitive data types are specified, as is the behavior of arithmetic on them. For example, "int" always means a signed two's complement 32 bit integer, and "float" always means a 32-bit IEEE 754 floating point number. Making these choices is feasible in this day and age because essentially all interesting CPUs share these characteristics.

The libraries that are a part of the system define portable interfaces. For example, there is an abstract Window class and implementations of it for Unix, Windows NT/95, and the Macintosh.

The Java system itself is quite portable. The compiler is written in Java and the runtime is written in ANSI C with a clean portability boundary. The portability boundary is essentially a POSIX subset.

## Interpreted

Java bytecodes are translated on the fly to native machine instructions (interpreted) and not stored anywhere. And since linking is a more incremental and lightweight process, the development process can be much more rapid and exploratory.

As a part of the bytecode stream, more compile-time information is carried over and available at runtime. This is what the linker's type checks are based on. It also makes programs more amenable to debugging.

The programmers at Archimedes spent a lot of time waiting for programs to compile and link. They also spent a lot of time tracking down senseless bugs because some changed source files didn't get compiled (despite using a fancy "make" facility), which caused version mismatches; and they had to track down procedures that were declared inconsistently in various parts of their programs. Another couple of months lost in the schedule.

## High Performance

While the performance of interpreted bytecodes is usually more than adequate, there are situations where higher performance is required. The bytecodes can be translated on the fly (at runtime) into machine code for the particular CPU the application is running on. For those accustomed to the normal design of a compiler and dynamic loader, this is somewhat like putting the final machine code generator in the dynamic loader.

The bytecode format was designed with generating machine codes in mind, so the actual process of generating machine code is generally simple. Efficient code is produced: the compiler does automatic register allocation and some optimization when it produces the bytecodes.

In interpreted code we're getting about 300,000 method calls per second on an Sun Microsystems SPARCStation 10. The performance of bytecodes converted to machine code is almost indistinguishable from native C or C++.

When Archimedes was starting up, they did a prototype in Smalltalk. This impressed the investors enough that they got funded, but it didn't really help them produce their product: in order to make their simulations fast enough and the system small enough, it had to be rewritten in C.

## Multithreaded

There are many things going on at the same time in the world around us. Multithreading is a way of building applications with multiple threads. Unfortunately, writing programs that deal with many things happening at once can be much more difficult than writing in the conventional single-threaded C and C++ style.

Java has a sophisticated set of synchronization primitives that are based on the widely used monitor and condition variable paradigm introduced by C.A.R.Hoare. By integrating these concepts into the language (rather than only in classes) they become much easier to use and are more robust. Much of the style of this integration came from Xerox's Cedar/Mesa system.

Other benefits of multithreading are better interactive responsiveness and real-time behavior. This is limited, however, by the underlying platform: stand-alone Java runtime environments have good real-time behavior. Running on top of other systems like Unix, Windows, the Macintosh, or Windows NT limits the real-time responsiveness to that of the underlying system.

Lots of things were going on at once in their simulations. Ropes were being pulled, wheels were turning, levers were rocking, and input from the user was being tracked. Because they had to write all this in a single threaded form, all the things that happen at the same time, even though they had nothing to do with each other, had to be manually intermixed. Using an "event loop" made things a little cleaner, but it was still a mess. The system became fragile and hard to understand. They were pulling in data from all over the net. But originally they were doing it one chunk at a time. This serialized network communication was very slow. When they converted to a multithreaded style, it was trivial to overlap all of their network communication.

## Dynamic

In a number of ways, Java is a more dynamic language than C or C++. It was designed to adapt to an evolving environment.

For example, one major problem with C++ in a production environment is a side-effect of the way that code is implemented. If company A produces a class library (a library of plug and play components) and company B buys it and uses it in their product, then if A changes its library and distributes a new release, B will almost certainly have to recompile and redistribute their own software. In an environment where the end user gets A and B's software independently (say A is an OS vendor and B is an application vendor) problems can result.

For example, if A distributes an upgrade to its libraries, then all of the software from B will break. It is possible to avoid this problem in C++, but it is extraordinarily difficult and it effectively means not using any of the language's OO features directly.

Archimedes built their product using the object-oriented graphics library from 3DPC Inc. 3DPC released a new version of the graphics library which several computer manufacturers bundled with their new machines. Customers of Archimedes that bought these new machines discovered to their dismay that their old software no longer worked. (In real life, backwards compatibility isn't always a high priority in the Unix world. In the PC world, 3DPC would never have released such a library: their ability to change their product and use C++'s object oriented features is severely hindered because they can't expect their customers to recompile.)

By making these interconnections between modules later, Java completely avoids these problems and makes the use of the object-oriented paradigm much more straightforward. Libraries can freely add new methods and instance variables without any effect on their clients.

An interface specifies a set of methods that an object can perform but leaves open how the object should implement those methods. A class implements an interface by implementing all the methods contained in the interface. In contrast, inheritance by subclassing passes both a set of methods and their implementations from superclass to subclass. A Java class can implement multiple interfaces but can only inherit from a single superclass. Interfaces promote flexibility and reusability in code by connecting objects in terms of what they can do rather than how they do it.

Classes have a runtime representation: there is a class named `Class`, instances of which contain runtime class definitions. If, in a C or C++ program, you have a pointer to an object but you don't know what type of object it is, there is no way to find out. However, in Java, finding out based on the runtime type information is straightforward. Because casts are checked at both compile-time and runtime, you can trust a cast in Java. On the other hand, in C and C++, the compiler just trusts that you're doing the right thing.

It is also possible to look up the definition of a class given a string containing its name. This means that you can compute a data type name and have it easily dynamically-linked into the running system.

To expand their revenue stream, the folks at Archimedes wanted to architect their product so that new aftermarket plug-in modules could be added to extend the system. This was possible on the PC, but just barely. They had to hire a couple of new programmers because it was so complicated. This also added problems when debugging.

## Summary

The Java language provides a powerful addition to the tools that programmers have at their disposal. Java makes programming easier because it is object-oriented and has automatic garbage collection. In addition, because compiled Java code is architecture-neutral, Java applications are ideal for a diverse environment like the Internet

## Java Platform

The computer world currently has many platforms, among them Microsoft Windows, Macintosh, OS/2, UNIX® and NetWare®; software must be compiled separately to run on each platform. The binary file for an application that runs on one platform cannot run on another platform, because the binary file is platform-specific.

The Java Platform is a new software platform for delivering and running highly interactive, dynamic, and secure applets and applications on networked computer systems. But what sets the Java Platform apart is that it sits on top of these other platforms, and executes *bytecodes*, which are not specific to any physical machine, but are machine instructions for a *virtual machine*. A program written in the Java Language compiles to a bytecode file that can run wherever the Java Platform is present, on *any* underlying operating system. In other words, the same exact file can run on any operating system that is running the Java Platform. This portability is possible because at the core of the Java Platform is the Java Virtual Machine.

While each underlying platform has its own implementation of the Java Virtual Machine, there is only one virtual machine specification. Because of this, the Java Platform can provide a standard, uniform programming interface to applets and applications on any hardware. The Java Platform is therefore ideal for the Internet, where one program should be capable of running on any computer in the world. The Java Platform is designed to provide this "Write Once, Run Anywhere"SM capability.

Developers use the Java Language to write source code for Java-powered applications. They compile once to the Java Platform, rather than to the underlying system. Java Language source code compiles to an intermediate, portable form of bytecodes that will run anywhere the Java Platform is present.

Developers can write object-oriented, multithreaded, dynamically linked applications using the Java Language. The platform has built-in security, exception handling, and automatic garbage collection. Just-in-time compilers are available to speed up execution by converting Java bytecodes into machine language. From within the Java Language, developers can also write and call native methods-methods in C, C++ or another language, compiled to a specific underlying operating system-for speed or special functionality.

The Java Language is the entry ramp to the Java Platform. Programs written in the Java Language and then compiled will run on the Java Platform. The Java Platform has two basic parts:

- Java Virtual Machine
- Java Application Programming Interface (Java API)

These are described in detail later in this paper. Combined, these parts provide an end-user runtime environment for deploying Internet and intranet applications.

## **The Java Base Platform**

The Java Base Platform is the *minimum* Java Platform that developers can safely assume is present for running Java-powered applets and applications. This platform applies to Network Computers, desktop computers, and workstations (the next section describes the platform for smaller systems). This platform contains the same Java Virtual Machine mentioned before, but has a minimal set of API required to run basic applets and applications. This minimal set is known as the Java Core API, also known as the Java Applet API or Java Base API. Developers who write to this minimum set can feel secure that the program will run anywhere without the need for additional class libraries.

Certain Java Platform licensees have contracted to include the Java Core API in their particular implementation of the Java Platform. As more class libraries are developed, the Java Base Platform will grow, and these additions will migrate in a timely fashion into the Java Base Platform present on each licensee's operating system.

Another set of APIs, called the Standard Extension API, is being defined by JavaSoft, in partnership with leading industry companies, to extend the base functionality. Over time, some subset of the Standard Extension API will migrate into the Java Base Platform.

## **The Embedded Java Platform**

The Embedded Java Platform is being targeted for consumer devices with fewer resources and more specialized functionality than a Network Computer, such as set-top boxes, printers, copiers, and cellular phones. Such devices might have special constraints such as small memory footprint, no display, or no connection to a network.

The API targeted for this platform is called the Java Embedded API. The Java Embedded API is the smallest API a low-function embedded device can have and still run. Because this platform is still under development, this API has not yet achieved the level of a standard. Consequently this API is not yet well-defined, but it will probably consist of the packages `java.lang` and `java.util`. A Java-powered application written for one particular device could operate on a wide range of similar, dedicated devices.

## **Benefits of the Java Platform**

The Java Platform has benefits for the end-user as well as the developer and support personnel:

### **End-User Benefits**

Today, the Java Platform provides live, interactive content on the World Wide Web, with just-in-time software access. Applications are readily available on all operating systems at once, freeing users from having to choose operating systems on that basis. Smaller, less expensive, dedicated systems will eventually be available for specialized applications.

## **Developer Benefits**

The Java Language is a small, "knowable" system and is coupled with a growingly comprehensive set of APIs. Developers can "Write Once, Run Anywhere," which provides tremendous marketing leverage over other languages. In addition, Java development environments on all operating systems compile to a single binary format. Rather than developing on multiple platforms to deliver on multiple platforms, developers can now develop on one platform, saving cost, to deliver on that same platform, which is everywhere. The ability to "Write Once, Run Anywhere" is enough reason for some developers to turn to the Java Language as an alternative to C or C++ even for stand-alone, non-networked applications.

In addition, building applications from shared, reusable objects can further reduce cost by allowing developers to concentrate on creating only what is novel. Developers can distribute by network rather than compete for shelf-space in software stores.

## **Administrative and Support Benefits**

The Java Platform has benefits for corporate computer systems administration departments. Version control and upgrades are simplified because Java-powered applications can be kept in a central repository and served from there for each individual use. In multivendor, multiplatform environments, the number of platforms to support is reduced to one. Emerging lower-cost network computers have the potential to reduce maintenance and capital expenditures. With these network computers, data management can remain centralized while data processing is done locally.

Companies with large intranets, that may not find it worthwhile to upgrade to the latest memory-consuming operating system, can run Java-powered applications on all their existing machines. By providing corporate data in a format readable by Java-powered applications, corporations give users the platform-neutral access to the data they need.

When customers are running on the Java Platform, companies can take advantage of the interactivity of the Internet by moving employee tasks out to customers. Companies can reduce time spent on order-entry by having customers fill in order-entry forms themselves on Web pages. This is more practical than previously possible, because the customer can now be on any operating system.

## **Applets and Applications**

The Java Platform enables developers to create two different kinds of programs:

**Applets** are programs that require a browser to run. The <applet> tag is embedded in a Web page and names the program to be run. When that page is accessed by a user, either over the Internet or corporate intranet, the applet automatically downloads from the server and runs on the client machine. Because applets are downloaded, they tend to be designed small or modular, to avoid large download times.

**Applications** are programs that do not require a browser to run—they have no built-in downloading mechanism. When an application is called, it runs. In this way, applications are just like programs in other languages. They can perform traditional desktop tasks, such as those done with a word processor, spreadsheet or graphics application. Like an applet, an application requires the Java Platform for it to run; however the Platform can be available as a separate program, can be embedded directly within the underlying operating system, or presumably can be embedded in the application itself.

While applets and applications have different means of being invoked, for the most part they have the same access to a wide range of language capabilities. For example, either an applet or an application can access a host database, retrieve the data it needs, do local data processing, and store the results back to the host.

However, an applet requires a network to run, while an application does not. Applications have greater freedom in that they have full access to system services. For example an application, unlike an applet, can have normal read and write access to files on any disk. Since an applet can potentially be downloaded from an untrusted Web page, it is restricted from having read or write access to any file system except the server from which it came. This constraint will be relaxed when applets can be marked with digital signatures, allowing the end-user to be assured that it has been downloaded unaltered from a trusted source. Where local file storage is required, currently an application is required.

## Where Will the Java Platform Be Deployed?

The progression towards ubiquity has great momentum, moving in three stages from browsers, to desktop, workstation and network operating systems, and finally to embedded devices.

First, the Java Base Platform is currently embedded in the top two most widely used Internet browsers, Netscape Navigator™, and Microsoft Internet Explorer. It is also available in HotJava™, from Sun Microsystems, and will become available in other browsers.

Second, the Java Base Platform will soon be embedded in all leading desktop, workstation, and network operating systems—see Figure 1. Being available on the combination of Microsoft Windows, Macintosh, OS/2, and UNIX computers, the Java Base Platform will have an installed base as large as these platforms combined. By targeting this platform, developers will tap into the new, exploding market for Web and intranet applications without being tied to any particular hardware or operating system environment. The Java Platform will become the platform for all network- and Web-based computing.

Third, with the Java Processors family of integrated circuits, the platform will be available in a wide range of consumer and industrial embedded devices such as dedicated Network Computers, set-top boxes, printers, copiers and cellular phones.

Operating Systems That Embed the Java Base Platform	
<b>Windows</b>	
• Microsoft Corporation	Windows 95, Windows NT
• International Business Machines	Windows 3.1
<b>Macintosh</b>	
• Apple Computer, Inc.	MacOS
<b>OS/2</b>	
• International Business Machines	OS/2
<b>Unix</b>	
• Hewlett Packard Corp.	HPUX
• Hitachi, Ltd.	Hitachi OS
• International Business Machines	AIX
• Silicon Graphics, Inc.	Irix
• SunSoft, Sun Microsystems, Inc.	Solaris™
• The Santa Cruz Operation, Inc. (SCO)	UnixWare
• Tandem Computers	Non-Stop Kernel
<b>Network OS</b>	
• Novell, Inc.	NetWare 4.0
<b>Maintframe OS</b>	
• International Business Machines	MVS

Figure 1. Companies licensing the Java Base Platform to embed in operating systems.

The Java promise is "write once, run anywhere". The first half of this promise is supplied by the Java programming language. As a language Java supports the needs of modern computing tasks with built-in features like threading, garbage collection and internationalization. Standard Java packages that are part of the JDK support even more with powerful networking classes, rich GUI frameworks and useful IO classes. Software developers need these features to create applications for today's marketplace.

The second half of the Java promise is supplied by the Java application environment, the component that allows a Java applet or application to execute on a specific system. A Java application environment includes a Java virtual machine and the necessary support packages and files. Java application environments are tailored to provide a good match between the Java programming language and the underlying system.

From the beginning, Java was designed to be scalable. To achieve this goal, application environments are needed for a variety of different systems. Java first gained notice with Internet browsers. Then came application environments for desktop systems like Windows 95/NT and UNIX. Now Java application environments are finding their way into network computers, consumer devices and embedded systems -- even smart cards where the Java application environment must fit into a tiny amount of memory!

## **Application-Hosted**

Today most major Internet browsers support Java. A browser is an example of an application-hosted Java application environment. It is sometimes called a \*container application\* because it contains a Java application environment that executes Java code. Some browsers have a Java application environment built-in while others use the Java application environment available in the underlying OS. Another example of a container application is a web server that supports Java servlets.

## **OS-Hosted**

Most common operating systems now offer a Java application environment that can execute Java applications. An important role of any Java application environment is to map system services and resources to the system-independent portion of the Java application environment. For example, if an operating system has a native window system or thread library, the Java application environment can use these directly. Providing access to these kinds of resources without affecting the essential portability of Java code is an important part of developing a Java application environment.

The degree to which the Java application environment is integrated into the underlying operating system varies. Early efforts to add Java to desktop operating systems provided minimal integration. This led to problems with application installation and execution as well as access to operating system resources. More recently Java application environments for desktop operating systems have provided better integration so that Java applications behave more like native applications.

The Java Runtime Environment is an example of an OS-Hosted Java application environment. The JRE can be downloaded from <http://www.javasoft.com> for use on Solaris or Win32 systems.

## **JavaOS<sup>TM</sup>**

JavaOS is a Java-centric operating system with a feature set and architecture designed to provide a complete Java environment for network computers. Network computers differ significantly from conventional stand-alone personal computers because they work entirely within a client-server environment. The main goal of the client-server computing model is to divide tasks by concentrating the user-interface on the client side and data management on the server side. JavaOS provides the operating system facilities needed to execute Java applets on the client side of this environment.

Because they are designed to provide only the user-interface portion of a client-server application, network computers offer the promise of low-cost computing devices with simplified client-side system administration. These two advantages significantly reduce the total cost of ownership for enterprise users.

JavaOS supports only 100% Pure Java applets running inside a Java-based container application like HotJava browser. The same Java applets that run in a web browser on a PC can run on JavaOS

on a network computer. These applets communicate through a network with applications like data base management systems running on a server.

Sun has licensed JavaOS to several manufacturers who will build JavaOS-based network computers that use a variety of different CPUs and peripherals. Future versions of network computers will use Java Processors.

## **JavaPC**

JavaPC is a version of JavaOS targeted at legacy PC's. It is a software product that turns a PC into a network computer. This is helpful for companies with an installed base of PC's that want to migrate towards a JavaOS-based network computer architecture. JavaPC provides a cost-effective way to enable this migration while protecting an investment in legacy PC's.

## **PersonalJava and EmbeddedJava**

PersonalJava and EmbeddedJava are Java application environments designed to run on top of real-time operating systems. PersonalJava is oriented towards communications-centric products like webtop devices and webphones. EmbeddedJava is designed to support high-volume embedded products that have severe resource limitations. PersonalJava and EmbeddedJava demonstrate Java's scalability by providing subsets of the core Java APIs.

## **Java Card**

Java Card brings the world of Java to smart cards, credit card sized devices with tiny embedded microprocessors. Smart cards provide a secure and programmable device for enabling digital commerce. Java Card includes a small Java virtual machine and a limited set of packages.

There are many benefits to basing smart cards on Java Card. From an implementation point of view the Java Card virtual machine and instruction set are a good match to the needs of a smart card. Since Java provides both security and communications support it is good for establishing a connection between a retail transaction and the banking system. Financial value and transaction records can be represented as Java objects on the smart card and conveniently transferred to other Java based devices.

## **Java Processors**

Improve Java performance dramatically in embedded applications. The picoJava<sup>TM</sup> CPU core is a licenseable technology that removes the need for a Java interpreter or JIT (Just-In-Time compiler) by directly executing Java bytecode into the silicon. This simplifies system design, improves performance, and reduces memory requirements. picoJava helps licensees develop PDAs, smartphones and other embedded consumer devices that run Java at its most cost-effective and efficient.

The picoJava processor core is optimized for Java and its unique demands, such as multithreading and garbage collection. Sun Microelectronics' microJava<sup>TM</sup>-701 processor prototype validates picoJava design and paves the way for licensees to produce their own Java-based microprocessors and devices. Evaluation boards for picoJava II, based on the 701 will soon be available for demonstration and design.

## **JavaOS<sup>TM</sup>**

JavaOS is an operating system that implements the Java Base Platform for running Java-powered applets and applications. As such, it implements the Java Virtual Machine, Java Embedded API, and the underlying functionality for windowing, networking and file system.

JavaOS is designed for Network Computers, consumer devices, and network devices for embedded applications, such as printers, copiers and industrial controllers. These devices will have instant

turn-on, no installation setup, no system administration, and, when on a network, can be automatically upgraded.

JavaOS will be widely ported to a range of microprocessors, including the Java Processors family. When JavaOS runs on a Java Processor, the microprocessor's silicon Java Virtual Machine is used.

## A Word About the Java Language

The Java Language is the means for a developer to write source code. Applets and applications written in the Java Language compile to a form that runs on the Java Platform.

When developers write source code in the Java Language, this code can call APIs defined in the Java Core API, Java Standard Extensions API, or a new API defined in the source code itself. At runtime, all three kinds of APIs have equal standing, and are not distinguished on the basis of their source. Compiling the source with the Java Compiler generates bytecodes that are executed on the Java Platform.

As professional programming languages go, the Java Language is simple, yet flexible and powerful. It is object-oriented (with single inheritance), statically typed, multithreaded, dynamically linked, and has automatic garbage collection.

Its syntax is based on C and C++, so those programmers can pick it up quite easily. There's less redundancy which means developers should be able to more easily read someone else's code. For example, the Java Language has no user-defined operator overloading, as is found in C++.

The Java Language gives developers the ability to do three different kinds of programming in one language. Like the symbolic programming language Smalltalk, the Java Language is object-oriented, has dynamic linking, and has a class hierarchy with single inheritance. For numeric programming, the Java Language has platform-independent data types, array bounds-checking, and well-defined IEEE arithmetic. These capabilities provide good grounding for writing stable numerical algorithms that give repeatable results. For systems programming, expressions, statements, and operators in the Java Language are in most cases the same as in the C language.

The Java Language encourages catching bugs early, during development, before the software is released. It does this by strong data typing, automatic garbage collection, array bounds checking, lack of automatic type coercion, and the lack of the pointer data type. These safeguards help in the age of the Internet, where developers are deploying software very rapidly.

The Java Language has multithreading built in, with a strong model of how thread-critical code can be synchronized to avoid race or timing problems. With the growth of multiprocessing and a decrease in processor costs, the Java Language is poised to enable a new generation of concurrent applications and services.

Exception and thread mechanisms are integrated with the language and its type system. In addition, the language includes dynamic linking of subclasses with methods that override or add functionality at runtime. In other environments, these features have typically been arcane and complicated system services. There is a great simplicity and advantage to having these facilities in the language and therefore portable between platforms. The language also defines what binary compatibility is, by defining a class (.class) file format, which includes the instructions for the Java Virtual Machine in the form of bytecodes.

## A Look Inside the Java Platform

The Java Platform has two main parts, the Java Virtual Machine and the Java API, as shown in Figure 2.

**Java Virtual Machine** - The Java Virtual Machine is a "soft" computer that can be implemented in software or hardware. It's an abstract machine designed to be implemented on top of existing

processors. The porting interface and adapters enable it to be easily ported to new operating systems without being completely rewritten.

**Java API** - The Java API forms a standard interface to applets and applications, regardless of the underlying operating system. The Java API is the essential framework for application development. This API specifies a set of essential interfaces in a growing number of key areas that developers will use to build their Java-powered applications.

**The Java Core API** provides the very basic language, utility, I/O, network, GUI, and applet services; OS companies that have licensed Java have contracted to include them in any Java Platform they deploy.

**The Java Standard Extension API** extends the capabilities of Java beyond the Java Core API. Some of these extensions will eventually migrate to the Java Core API. Other nonstandard extension APIs can be provided by the applet, application, or underlying operating system. As each new extension API specification is published, it will be made available for industry review and feedback before it is finalized.

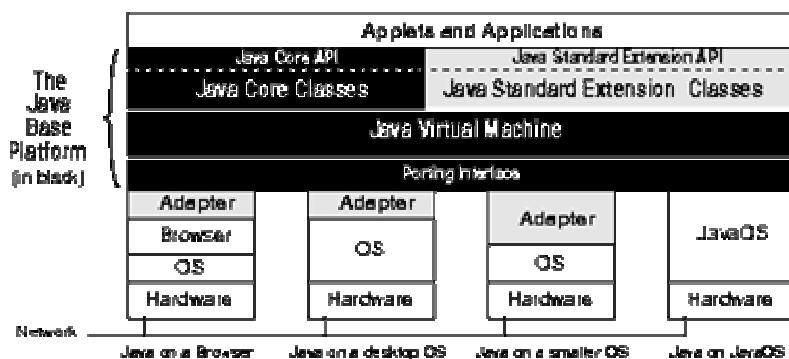


Figure 2. The Java Base Platform is uniform across all operating systems.

In this figure, the Java Base Platform is the part shown in black, including the blocks labeled Adapter. The Java API includes both the Java Core API and Java Standard Extension API. The classes are the implementation of that API. The Java Virtual Machine is at the core of the platform. The Porting Interface lies between the Java Virtual Machine and the operating system (OS) or browser. This Porting Interface has a platform independent part (shown in black) and a platform-dependent part, shown as Adapters. The OS and JavaOS provide the window, filing and network functionality. Different machines can be connected by a network, as shown.

The Java API framework is open and extensible. Specifications for each interface are being developed by industry-wide specialists in each area. Forthcoming specifications will be published and open to industry review. Implementations of the API specifications will be available from JavaSoft and others throughout the industry. In today's environment of rapid innovation, the Java API framework allows that innovation to easily exist as extensions to the Java Platform.

The API is organized by groups, or sets. Each of the API sets can be implemented as one or more packages (namespaces). Each package groups together a set of classes and interfaces that define a set of related fields, constructors, and methods.

## Java Virtual Machine

The Java Virtual Machine is key to the independence of the underlying operating system and hardware-it is a platform that hides the underlying operating system from Java-powered applets and applications. And it's very easy to port the Virtual Machine to a browser or another operating system.

In addition, the Virtual Machine defines a machine-independent format for binary files called the class (.class) file format. This format includes instructions for a virtual computer in the form of bytecodes. The bytecode representation of any Java Language program is symbolic in the sense that offsets and indexes into methods are not constants, but are, instead, given symbolically as string names. The first time a method is called, it is searched by name in the class file format, and its offset numeric value is determined at that time for quicker access at subsequent lookups. Therefore, any new or overriding method can be introduced late at runtime anywhere in the class structure, and it will be referred to symbolically, and properly accessed without breaking the code.

The bytecodes are a high-level representation of the program so that optimization and machine code generation (via a just-in-time compiler) can be performed at that level. In addition, garbage collection can occur inside the Virtual Machine, since it holds variables in stacks in the Java Platform address space.

## **Java Core API**

By targeting the APIs in the Java Platform, developers are assured that their applications will run everywhere for the single possible exception-dedicated, constrained, embedded systems, such as set-top boxes, printers, copiers, and cellular phones.)

The Java Core API defines the basic building blocks for creating fully functional Java-powered applets and applications. It includes all the classes in the java package: java.lang, java.util, java.io, java.net, java.awt, and java.applet. (Notice that the Java Core API is currently the full set of APIs available in version 1.0.2 of the Java Development Kit from JavaSoft.)

Over time, as the platform develops, this base will grow, both as we develop new core technology, and as some of the Standard Extension API migrate into the Java Core API.

## **Java Standard Extension API**

This section describes the Java Standard Extension API. These extensions are "standard" in that they form a published, uniform, open API that anyone can implement. Once defined, they can be added to, but, to maintain backward compatibility, not changed in a way that calls to them would fail. Over time, new extensions will be added. In addition, some of these extensions will migrate into the Java Core API.

The current plan for this migration is shown in the following table:

<b>Migrate to</b>	<b>Remain as</b>
<b>Java Core API</b>	<b>Java Standard Extension</b>
Java 2D	Java 3D
Audio	Video, MIDI
Java Media Framework	Java Share
Java Animation	Java Telephony
Java Enterprise	Java Server
Java Commerce	Java Management
Java Security	

An implementation should do whatever is appropriate for the platform it runs on, within its hardware constraints. For example, desktop operating systems that have access to a speaker will produce audio; however, a mainframe or network operating system that has no speaker is permitted to do the equivalent of a no-op or some other well-defined behavior, such as throw an exception.

## **Java Security API**

The Java Security API is a framework for developers to easily and securely include security functionality in their applets and applications. This functionality includes cryptography, with digital signatures, encryption and authentication.

Java Security includes an abstract layer that applications can call. That layer, in turn, makes calls to Java Security packages that implement the actual cryptography. This allows third-party developers specializing in providing cryptographic functionality to write packages for Java Security. Java Security also includes system support for key management, including a secure database, certificate facilities, and so on.

This architecture provides for replaceable, upgradable security. Whenever a stronger algorithm or a faster implementation becomes available, modules can be replaced in the platform, in a manner completely transparent to applications.

## **Java Media API**

The Java Media API defines the multimedia classes that support a wide range of rich, interactive media on and off the Web, including audio, video, 2D, 3D, animation, telephony, and collaboration. An extensible Media Framework provides for common control and synchronization of all time-based media (audio, video, animation, video teleconferencing) as well as for filters and processors.

The Java Media API is composed of several distinct components, each associated with either a specific type of media (audio, video, 2D, 3D), or a media-related activity (animation, collaboration, telephony). Collectively, these interfaces provide Java Language programmers with the ability to handle a wide variety of different media types within their applications and applets.

The Java Media API is highly extensible. The API accommodates today's large and ever-changing suite of media transports, containers, and encoding formats, and allows the addition of new media-related functionality as they become available.

JavaSoft has been working with a group of industry-leading companies to establish the standards for Java Media: Adobe®, Apple, Intel, Macromedia, Netscape, SGI, and Sun Microsystems.

The components of the Java Media APIs are as follows.

**Java 2D API** - Provides graphics and imaging capabilities beyond those available with the Java Core API. The 2D API allows the creation of high-quality, platform-independent graphics including line art, text, and images in a single model that uniformly addresses color, spatial transforms and compositing. It also provides an extension mechanism to support a wide array of different presentation devices (such as displays and printers), image formats, image encodings, color spaces, and compositors.

**Java Media Framework API** - Handles traditional time-critical media, such as audio, video and MIDI. The framework provides a common model for timing, synchronization, and composition, which can be applied to media components to allow them to interoperate. It is designed to handle streaming data, live or stored, compressed or raw, as well as from sampled audio and video streams.

**Video API** - Accommodates both streaming and stored video sources. It defines basic data formats and control interfaces.

**Audio API** - Supports sampled and synthesized audio. It includes a specification for 3D spatial audio, and accommodates both streaming and stored audio sources.

**MIDI API** - Provides support for timed-event streams. It uses the Media Framework for synchronization with other activities, and for an extensibility mechanism for new synthesizers and effects.

**Java Animation API** - Supports traditional 2D animation of sprites, with stacking order control. It makes use of 2D interfaces for compositing and the Media Framework for synchronization, composition, and timing.

**Java Share API** - Provides the basic abstraction for live, two-way, multi-party communication between objects over a variety of networks and transport protocols. The API enables synchronization and session management, and allows sharing of both "collaboration-aware" and "collaboration-unaware" applets.

**Java Telephony API** - Unifies computer/telephony integration. It provides basic functionality for control of phone calls: 1st-party call control (simple desktop phone), 3rd-party call control (phone call distribution center), teleconferencing, call transfer, caller ID, and DTMF decode/encode.

**Java 3D API** - Provides high-performance, interactive, 3D graphics support. It supports VRML, and has a high-level specification of behavior and control of 3D objects. The 3D API simplifies 3D application programming and provides access to lower level interfaces for performance. The 3D API is closely integrated with Audio, Video, MIDI, and Animation areas.

## Java Enterprise API

The Enterprise classes connect Java-powered applications to enterprise information resources. There are currently three groups for connectivity: JDBC™ (Java Database Connectivity), Interface Definition Language, and Remote Method Invocation.

**JDBC™** (Java Database Connectivity) is a standard SQL database access interface. It provides Java programmers with a uniform interface to a wide range of relational databases, and also provides a common base on which higher level tools and interfaces can be built. Partners such as Intersolv, Visigenic, and a dozen other database-connectivity vendors are providing JDBC drivers in the next few months for dozens of DBMSs, including Oracle, Sybase, and Informix. These database companies, and leading tool vendors, such as Symantec and Borland, have already endorsed the JDBC API and are developing products using JDBC.

**The JDBC API** defines classes to represent constructs such as database connections, SQL statements, result sets, and database metadata. JDBC allows a Java-powered program to issue SQL statements and process the results.

In conjunction with JDBC, JavaSoft is releasing a JDBC-ODBC bridge implementation that allows any of the dozens of existing Microsoft ODBC database drivers to operate as JDBC drivers. The JDBC-ODBC bridge can run on the server rather than client side using a JDBC driver that translates to a DBMS-independent network protocol.

**Interface Definition Language (IDL)** is a language-neutral way to specify an interface between an object and its client when they are on different platforms. This IDL component provides a mapping of its methods, packages, and other features to IDL operations and features.

**Remote Method Invocation (RMI)** lets programmers create Java objects whose methods can be invoked from another virtual machine. RMI is analogous to a remote procedure call (RPC) in the non-object world.

## **Java Commerce API**

The Java Commerce API brings secure purchasing and financial management to the Web.

World wide purchases of retail and wholesale goods and services during 1994 totaled 4.6 trillion dollars, 13% of which was spent remotely via catalogs, television, and various public and private communications networks. As this remote commerce migrates to the Internet, a standard framework within which to conduct these transactions becomes increasingly important. This is especially true as a rapidly increasing number of players vie for position to provide such facilities as electronic currency, online shopping malls, digital signature verification, and financial analysis.

The initial component of the Java Commerce API is the Java Wallet, which defines and implements a client-side framework for conducting network-based commerce. Think of Java Wallet as an empty wallet ready to hold credit cards and cash, and a blank ID card ready to be filled in with personal information.

The Java Wallet provides:

- Storage of personal information about:
  - The shopper (such as name, billing address, and shipping address)
  - Payment instruments (such as credit cards, debit cards, and electronic cash)
  - Details of every purchase transaction (such as date and time, item descriptions, quantities, and dollar amounts)
- Support for two new types of signed applets
  - Payment cassettes, which implement a specific payment protocol, such as the Secure Electronic Transaction (SET) supported by Visa and MasterCard
  - Service cassettes, which implement value-added services, such as budgeting and financial analysis
- Extensibility to install new payment and service cassettes dynamically by downloading them from the network
- Strong cryptographic services that make it possible to implement all of the facilities described above in a secure environment

## **Java Server API**

Java Server is an extensible framework that enables and eases the development of a whole spectrum of Java-powered Internet and intranet servers. The framework API contains server-side class libraries for server administration, access control, and dynamic server resource handling. The framework also encompasses the Servlet API.

Servlets are platform-independent Java-powered objects, the server side counterpart of applets. Servlets can reside locally on the server, or be downloaded to the server from the net under security restrictions. Servlet examples range from simple HTTP servlets (efficient replacements of cgi-scripts) to more complex servlets using JDBC/ODBC that offer database connectivity.

## **Java Management API**

The Java Management API is a collection of Java Classes that provide the building blocks for integrated management. It does this by providing a number of interfaces, classes, applets, and guidelines that facilitate the development of integrated management solutions.

The Java Management API is composed of several distinct components, each associated with an aspect of the total management space. Taken together the objects defined using the Java Management API will encapsulate distributed network, system, and service management components.

The components of the Java Management APIs are as follows:

**Admin View Module** - The Admin View Module is an extension of the Java Abstract Window Toolkit (AWT) that is specifically designed for creating integrated management solutions. The Admin View Module classes are used to implement the user model that builds on the web browser hypertext style of navigation.

**Base Object Interfaces** - The Base Object Interfaces support constructing objects that represent distributed resources and services that make up the enterprise computing environment. These interfaces allow developers to define abstractions that contain distributed attributes and methods, and persistent attributes.

**Managed Notification Interfaces** - The Managed Notification Interfaces provide the basic foundation from which more complex event-management services can be easily built. The model provides asynchronous event notification between managed objects or management applications providing the interfaces to an implementation of a basic event-dispatching service.

**Managed Container Interfaces** - These allow managed objects to be grouped so that management applications can perform actions on a single group, instead of each instance. This permits management applications to scale upwards by allowing for multiple instances to be treated as one. Container interfaces are of two types: *Extensional*, which are constructed programmatically through simple add and remove methods, and *Intentional*, which store only the query to be executed-by using the Managed Data Interfaces-that generate the instances for the container.

**Managed Data Interfaces** - These support mapping attributes of extensions to the Base Object Interfaces to a relational database. These interfaces are implemented on the appropriate subset of JDBC (Java Database Connectivity). These interfaces support a number of commercially available relational database engines.

**Managed Protocol Interfaces** - These implement the distribution and security capabilities for extensions of the Base Object Interfaces. These interfaces build up the Java Security APIs and Java Remote Method Invocation (RMI).

**SNMP Interfaces** - These extend the Managed Protocol Interfaces to allow extensions of the Base Objects to contain information obtained from existing SNMP agents. By extending the Managed Protocol Interfaces, this allows SNMP information to be available to all users of the Java Management API.

The "Java Management API User Interface Style Guide" provides guidelines for developing interfaces for configuration and troubleshooting of the system, network, and service elements that make up the computing infrastructure.

## **Java Compile and Runtime Environments**

A Java Language development environment includes both the compile-time and runtime environments, as shown in Figure 3. The Java Platform is represented by the runtime environment.

The developer writes Java Language source code (.java files) and compiles it to bytecodes (.class files). These bytecodes are instructions for the Java Virtual Machine. To create an applet, the developer next stores these bytecode files on an HTTP server, and adds an <applet code=filename> tag to a Web page, which names the entry-point bytecode file.

When an end user visits that page, the <applet> tag causes the bytecode files to be transported over the network from the server to the end user's browser in the Java Platform. At this end, the bytecodes are loaded into memory and then verified for security before they enter the Virtual Machine.

Once in the Virtual Machine, the bytecodes are interpreted by the Interpreter, or optionally turned into machine code by the just-in-time (JIT) code generator, known more commonly as the JIT Compiler. The Interpreter and JIT Compiler operate in the context of the runtime system (threads, memory, other system resources). Any classes from the Java Class Libraries (API) are dynamically loaded as needed by the applet.

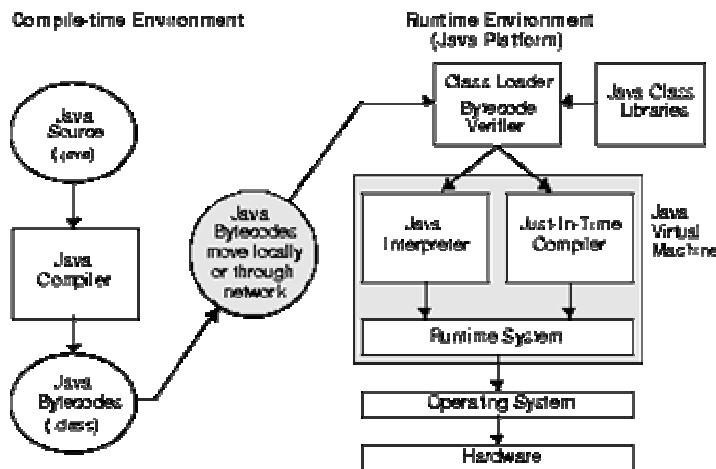


Figure 3. Source code is compiled to bytecodes, which are executed at runtime.

# PHP (PHP: Hypertext Preprocessor)

## Introduction

PHP ("PHP: Hypertext Preprocessor") is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

## An introductory example

```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <?php
      echo "Hi, I'm a PHP script!";
    ?>
  </body>
</html>
```

Notice how this is different from a script written in other languages like Perl or C -- instead of writing a program with lots of commands to output HTML, you write an HTML script with some embedded code to do something (in this case, output some text). The PHP code is enclosed in special start and end tags that allow you to jump into and out of "PHP mode".

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server. If you were to have a script similar to the above on your server, the client would receive the results of running that script, with no way of determining what the underlying code may be. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

The best things in using PHP are that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer.

## What can PHP do?

PHP is mainly focused on server-side scripting, so you can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies. But PHP can do much more.

There are three main areas where PHP scripts are used.

- **Server-side scripting.** This is the most traditional and main target field for PHP. You need three things to make this work. The PHP parser (CGI or server module), a web server and a web browser. You need to run the web server, with a connected PHP installation. You can access the PHP program output with a web browser, viewing the PHP page through the server.
- **Command line scripting.** You can make a PHP script to run it without any server or browser. You only need the PHP parser to use it this way.
- **Writing desktop applications.** PHP is probably not the very best language to create a desktop application with a graphical user interface, but if you know PHP very well, and would like to use some advanced PHP features in your client-side applications you can also use PHP-GTK to write such programs. You also have the ability to write cross-platform

applications this way. PHP-GTK is an extension to PHP, not available in the main distribution.

PHP can be used on all major operating systems, including Linux, many Unix variants (including HP-UX, Solaris and Open BSD), Microsoft Windows, Mac OS X, RISC OS, and probably others. PHP has also support for most of the web servers today. This includes Apache, Microsoft Internet Information Server, Personal Web Server, Netscape and iPlanet servers, O'Reilly Website Pro server, Caudium, Xitami, OmniHTTPD, and many others. For the majority of the servers PHP has a module, for the others supporting the CGI standard, PHP can work as a CGI processor.

So with PHP, you have the freedom of choosing an operating system and a web server. Furthermore, you also have the choice of using procedural programming or object oriented programming, or a mixture of them. Although not every standard OOP feature is implemented in PHP 4, many code libraries and large applications (including the PEAR library) are written only using OOP code. PHP 5 fixes the OOP related weaknesses of PHP 4, and introduces a complete object model.

One of the strongest and most significant features in PHP is its support for a wide range of databases. Writing a database-enabled web page is incredibly simple. The following databases are currently supported:

Adabas D	InterBase	PostgreSQL
dBase	FrontBase	SQLite
Empress	mSQL	Solid
FilePro (read-only)	Direct MS-SQL	Sybase
Hyperwave	MySQL	Velocis
IBM DB2	ODBC	Unix dbm
Informix	Oracle (OCI7 and OCI8)	
Ingres	Ovrimos	

PHP also has support for talking to other services using protocols such as LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (on Windows) and countless others. You can also open raw network sockets and interact using any other protocol. PHP has support for the WDDX complex data exchange between virtually all Web programming languages. Talking about interconnection, PHP has support for instantiation of Java objects and using them transparently as PHP objects. You can also use our CORBA extension to access remote objects.

## Your first PHP-enabled page

**Create a file named hello.php and put it in your web server's root directory with the following content:**

```
<html>
<head>
<title>PHP Test</title>
</head>
<body>
<?php echo '<p>Hello World</p>' ; ?>
</body>
</html>
```

Use your browser to access the file with your web server's URL, ending with the "/hello.php" file reference. When developing locally this URL will be something like <http://localhost/hello.php> or <http://127.0.0.1/hello.php> but this depends on the web server's configuration. If everything is configured correctly, this file will be parsed by PHP and the following output will be sent to your browser:

```

<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <p>Hello World</p>
  </body>
</html>

```

This program is extremely simple and you really did not need to use PHP to create a page like this. All it does is display: *Hello World* using the PHP echo() statement. Note that the file does not need to be executable or special in any way. The server finds out that this file needs to be interpreted by PHP because you used the ".php" extension, which the server is configured to pass on to PHP.

The point of the example is to show the special PHP tag format. In this example we used `<?php` to indicate the start of a PHP tag. Then we put the PHP statement and left PHP mode by adding the closing tag, `?>`. You may jump in and out of PHP mode in an HTML file like this anywhere you want.

**A Note on Text Editors:** There are many text editors and Integrated Development Environments (IDEs) that you can use to create, edit and manage PHP files. Having an editor with syntax highlighting can be helpful.

**A Note on Windows Notepad:** If you are writing your PHP scripts using Windows Notepad, you will need to ensure that your files are saved with the .php extension. (Notepad adds a .txt extension to files automatically unless you take one of the following steps to prevent it.) When you save the file and are prompted to provide a name for the file, place the filename in quotes (i.e. "hello.php"). Alternatively, you can click on the 'Text Documents' drop-down menu in the 'Save' dialog box and change the setting to "All Files". You can then enter your filename without quotes.

## Basic syntax

When PHP parses a file, it looks for opening and closing tags, which tell PHP to start and stop interpreting the code between them. Parsing in this manner allows php to be embedded in all sorts of different documents, as the PHP parser ignores everything outside of a pair of opening and closing tags. Most of the time you will see php embedded in HTML documents, as in this example.

```

<p>This is going to be ignored.</p>
<?php echo 'While this is going to be parsed.' ; ?>
<p>This will also be ignored.</p>

```

You can also use more advanced structures:

### Example 1. Advanced escaping

```

<?php
if ($expression) {
  ?>
    <strong>This is true.</strong>
  <?php
} else {
  ?>
    <strong>This is false.</strong>
  <?php
}

```

```
?>
```

This works as expected, because when PHP hits the ?> closing tags, it simply starts outputting whatever it finds until it hits another opening tag. The example given here is contrived, of course, but for outputting large blocks of text, dropping out of PHP parsing mode is generally more efficient than sending all of the text through **echo()** or **print()**.

There are four different pairs of opening and closing tags which can be used in php. Two of those, <?php ?> and <script language="php"> </script>, are always available. The other two are short tags and ASP style tags, and can be turned on and off from the php.ini configuration file. As such, while some people find short tags and ASP style tags convenient, they are less portable, and generally not recommended.

Note: Also note that if you are embedding PHP within XML or XHTML you will need to use the <?php ?> tags to remain compliant with standards.

## Example 2. PHP Opening and Closing Tags

1. <?php echo 'if you want to serve XHTML or XML documents, do like this'; ?>
2. <script language="php">  
 echo 'some editors (like FrontPage) don\'t  
 like processing instructions';  
</script>
3. <? echo 'this is the simplest, an SGML processing instruction'; ?>  
<?= expression ?> This is a shortcut for "<? echo expression ?>"
4. <% echo 'You may optionally use ASP-style tags'; %>  
<%= \$variable; # This is a shortcut for "<% echo . . ." %>

While the tags seen in examples one and two are both always available, example one is the most commonly used, and recommended, of the two.

Note: Support for ASP tags was added in 3.0.4.

## Instruction separation

As in C or Perl, PHP requires instructions to be terminated with a semicolon at the end of each statement. The closing tag of a block of PHP code automatically implies a semicolon; you do not need to have a semicolon terminating the last line of a PHP block. The closing tag for the block will include the immediately trailing new line if one is present.

```
<?php  
    echo 'This is a test';  
?  
<?php echo 'This is a test' ?>  
<?php echo 'We omitted the last closing tag';
```

Note: The closing tag of a PHP block at the end of a file is optional, and in some cases omitting it is helpful when using **include()** or **require()**, so unwanted white space will not occur at the end of files, and you will still be able to add headers to the response later. It is also handy if you use output buffering, and would not like to see added unwanted white space at the end of the parts generated by the included files.

## Comments

PHP supports 'C', 'C++' and Unix shell-style (Perl style) comments. For example:

```
<?php
    echo 'This is a test'; // This is a one-line c++ style
comment
    /* This is a multi line comment
       yet another line of comment */
    echo 'This is yet another test';
    echo 'One Final Test'; # This is a one-line shell-style
comment
?>
```

The "one-line" comment styles only comment to the end of the line or the current block of PHP code, whichever comes first. This means that HTML code after //...?> or #...?> WILL be printed: ?> breaks out of PHP mode and returns to HTML mode, and // or # cannot influence that. If the asp tags configuration directive is enabled, it behaves the same with // %> and # %>. However, the </script> tag doesn't break out of PHP mode in a one-line comment.

```
<h1>This is an <?php # echo 'simple';?> example.</h1>
<p>The header above will say 'This is an example'.</p>
```

'C' style comments end at the first \*/ encountered. Make sure you don't nest 'C' style comments. It is easy to make this mistake if you are trying to comment out a large block of code.

```
<?php
/*
    echo 'This is a test'; /* This comment will cause a problem
*/
*/
?>
```

## Data Types

PHP supports eight primitive types.

Four scalar types:

- **boolean**
- **integer**
- **float** (floating-point number, aka '**double**')
- **string**

Two compound types:

- **array**
- **object**

And finally two special types:

- **resource**
- **NULL**

This manual also introduces some pseudo-types for readability reasons:

- **mixed**
- **number**
- **callback**

You may also find some references to the type "double". Consider double the same as float, the two names exist only for historic reasons.

The type of a variable is usually not set by the programmer; rather, it is decided at runtime by PHP depending on the context in which that variable is used.

Note: If you want to check out the type and value of a certain expression, use **var\_dump()**.

**Note:** If you simply want a human-readable representation of the type for debugging, use **gettype()**. To check for a certain type, do not use **gettype()**, but use the *is\_type* functions. Some examples:

```
<?php
$a_bool = TRUE;      // a boolean
$a_str  = "foo";    // a string
$a_str2 = 'foo';    // a string
$an_int = 12;        // an integer

echo gettype($a_bool); // prints out: boolean
echo gettype($a_str); // prints out: string

// If this is an integer, increment it by four
if (is_int($an_int)) {
    $an_int += 4;
}

// If $bool is a string, print it out
// (does not print out anything)
if (is_string($a_bool)) {
    echo "String: $a_bool";
```

```
}
```

If you would like to force a variable to be converted to a certain type, you may either cast the variable or use the **settype()** function on it.

Note that a variable may be evaluated with different values in certain situations, depending on what type it is at the time.

## Booleans

This is the easiest type. A boolean expresses a truth value. It can be either **TRUE** or **FALSE**.

**Note:** The boolean type was introduced in PHP 4.

## Syntax

To specify a boolean literal, use either the keyword **TRUE** or **FALSE**. Both are case-insensitive.

```
<?php
$foo = True; // assign the value TRUE to $foo
?>
```

Usually you use some kind of operator which returns a Boolean value, and then pass it on to a control structure.

```
<?php
// == is an operator which test
// equality and returns a boolean
if ($action == "show_version") {
    echo "The version is 1.23";
}

// this is not necessary...
if ($show_separators == TRUE) {
    echo "<hr>\n";
}

// ...because you can simply type
if ($show_separators) {
    echo "<hr>\n";
}
?>
```

## Converting to boolean

To explicitly convert a value to boolean , use either the *(bool)* or the *(boolean)* cast. However, in most cases you do not need to use the cast, since a value will be automatically converted if an operator, function or control structure requires a boolean argument.

When converting to boolean ,the following values are considered **FALSE**:

- the boolean **FALSE** itself
- the integer 0 (zero)
- the float 0.0 (zero)
- the empty string, and the string "0"
- an array with zero elements
- an object with zero member variables (PHP 4 only)
- the special type **NULL** (including unset variables)
- Simple XML objects created from empty tags

Every other value is considered **TRUE** (including any resource).

<b>Warning</b>
-1 is considered <b>TRUE</b> , like any other non-zero (whether negative or positive) number!

```
<?php
var_dump((bool) "");           //
bool(false)
var_dump((bool) 1);            //
bool(true)
var_dump((bool) -2);          //
bool(true)
var_dump((bool) "foo");        //
bool(true)
var_dump((bool) 2.3e5);        //
bool(true)
var_dump((bool) array(12));   //
bool(true)
var_dump((bool) array());      //
bool(false)
var_dump((bool) "false");      //
bool(true)
?>
```

## **I**ntegers

An **integer** is a number of the set  $Z = \{\dots, -2, -1, 0, 1, 2, \dots\}$ .

### Syntax

Integers can be specified in decimal (10-based), hexadecimal (16-based) or octal (8-based) notation, optionally preceded by a sign (- or +).

If you use the octal notation, you must precede the number with a *O* (zero), to use hexadecimal notation precede the number with *0x*.

### Example 1. Integer literals

```
<?php
$a = 1234; // decimal number
$a = -123; // a negative number
$a = 0123; // octal number (equivalent to 83 decimal)
$a = 0x1A; // hexadecimal number (equivalent to 26 decimal)

?>
```

Formally the possible structure for integer literals is:

```
decimal      : [1-9][0-9]*
| 0

hexadecimal : 0[xX][0-9a-fA-F]++

octal        : 0[0-7]++

integer      : [+ -]?decimal
| [+ -]?hexadecimal
| [+ -]?octal
```

The size of an integer is platform-dependent, although a maximum value of about two billion is the usual value (that's 32 bits signed). PHP does not support unsigned integers.

## Integer overflow

If you specify a number beyond the bounds of the **integer** type, it will be interpreted as a **float** instead. Also, if you perform an operation that results in a number beyond the bounds of the **integer** type, a **float** will be returned instead.

There is no integer division operator in PHP. `1/2` yields the **float** `0.5`. You can cast the value to an integer to always round it downwards, or you can use the **round()** function.

```
<?php
var_dump(25/7);           // float(3.5714285714286)
var_dump((int)(25/7));    // int(3)
var_dump(round(25/7));    // float(4)
?>
```

## Converting to integer

To explicitly convert a value to **integer**, use either the `(int)` or the `(integer)` cast. However, in most cases you do not need to use the cast, since a value will be automatically converted if an operator, function or control structure requires an **integer** argument. You can also convert a value to integer with the function **intval()**.

### From booleans

`FALSE` will yield `0` (zero), and `TRUE` will yield `1` (one).

### From floating point numbers

When converting from float to integer, the number will be rounded towards zero.

If the float is beyond the boundaries of integer (usually  $+/- 2.15e+9 = 2^{31}$ ), the result is undefined, since the float hasn't got enough precision to give an exact integer result. No warning, not even a notice will be issued in this case!

```
<?php  
echo (int) ( (0.1+0.7) * 10 ); // echoes 7!  
?>
```

## From strings

See String conversion to numbers

## From other types

### Caution

Behaviour of converting to integer is undefined for other types. Currently, the behaviour is the same as if the value was first converted to boolean. However, do not rely on this behaviour, as it can change without notice.

## Floating point numbers

Floating point numbers (AKA "floats", "doubles" or "real numbers") can be specified using any of the following syntaxes:

```
<?php  
$a = 1.234;  
$b = 1.2e3;  
$c = 7E-10;  
?>
```

Formally:

```
LNUM          [0-9]+  
DNUM          ([0-9]*[\.] {LNUM}) | ({LNUM} [\.] [0-9]*)  
EXPONENT_DNUM ( ({LNUM}) | {DNUM}) [eE][+-]? {LNUM})
```

The size of a float is platform-dependent, although a maximum of ~1.8e308 with a precision of roughly 14 decimal digits is a common value (that's 64 bit IEEE format).

## Strings

A **string** is series of characters. In PHP, a character is the same as a byte, that is, there are exactly 256 different characters possible. This also implies that PHP has no native support of Unicode. See **utf8\_encode()** and **utf8\_decode()** for some Unicode support.

**Note:** It is no problem for a string to become very large. There is no practical bound to the size of strings imposed by PHP, so there is no reason at all to worry about long strings.

## Syntax

A string literal can be specified in three different ways.

- single quoted
- double quoted
- heredoc syntax

## Single quoted

The easiest way to specify a simple string is to enclose it in single quotes (the character ').

To specify a literal single quote, you will need to escape it with a backslash (\), like in many other languages. If a backslash needs to occur before a single quote or at the end of the string, you need to double it. Note that if you try to escape any other character, the backslash will also be printed! So usually there is no need to escape the backslash itself.

Note: In PHP 3, a warning will be issued at the *E\_NOTICE* level when this happens.

Note: Unlike the two other syntaxes, variables and escape sequences for special characters will not be expanded when they occur in single quoted strings.

```
<?php
echo 'this is a simple string';

echo 'You can also have embedded newlines in
strings this way as it is
okay to do';

// Outputs: Arnold once said: "I'll be back"
echo 'Arnold once said: "I\'ll be back"'; 

// Outputs: You deleted C:\*.*?
echo 'You deleted C:\\\*.*?'; 

// Outputs: You deleted C:\*.*?
echo 'You deleted C:\*.*?'; 

// Outputs: This will not expand: \n a newline
echo 'This will not expand: \n a newline';

// Outputs: Variables do not $expand $either
echo 'Variables do not $expand $either';
?>
```

## Double quoted

If the string is enclosed in double-quotes ("), PHP understands more escape sequences for special characters:

**Table 1. Escaped characters**

sequence	meaning
\n	linefeed (LF or 0x0A (10) in ASCII)
\r	carriage return (CR or 0x0D (13) in ASCII)
\t	horizontal tab (HT or 0x09 (9) in ASCII)

sequence	meaning
\\	backslash
\\$	dollar sign
\"	double-quote
\{0-7\}{1,3}	the sequence of characters matching the regular expression is a character in octal notation
\x{0-9A-Faf}{1,2}	the sequence of characters matching the regular expression is a character in hexadecimal notation

Again, if you try to escape any other character, the backslash will be printed too! Before PHP 5.1.1, backslash in \{\$var} hasn't been printed.

But the most important feature of double-quoted strings is the fact that variable names will be expanded.

## Heredoc

Another way to delimit strings is by using heredoc syntax ("<<<"). One should provide an identifier after <<<, then the string, and then the same identifier to close the quotation.

The closing identifier must begin in the first column of the line. Also, the identifier used must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

Heredoc text behaves just like a double-quoted string, without the double-quotes. This means that you do not need to escape quotes in your here docs, but you can still use the escape codes listed above. Variables are expanded, but the same care must be taken when expressing complex variables inside a heredoc as with strings.

### Example 2. Heredoc string quoting example

```
<?php
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* More complex example, with variables. */
class foo
{
    var $foo;
    var $bar;

    function foo()
    {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
```

```

$name = 'MyName';

echo <<<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
?>

```

**Note:** Heredoc support was added in PHP 4.

## Variable parsing

When a string is specified in double quotes or with heredoc, variables are parsed within it.

There are two types of syntax: a simple one and a complex one. The simple syntax is the most common and convenient. It provides a way to parse a variable, an array value, or an object property.

The complex syntax was introduced in PHP 4, and can be recognised by the curly braces surrounding the expression.

### Simple syntax

If a dollar sign (\$) is encountered, the parser will greedily take as many tokens as possible to form a valid variable name. Enclose the variable name in curly braces if you want to explicitly specify the end of the name.

```

<?php
$beer = 'Heineken';
echo "$beer's taste is great"; // works, '' is an invalid
character for varnames
echo "He drank some $beers"; // won't work, 's' is a valid
character for varnames
echo "He drank some ${beer}s"; // works
echo "He drank some {$beer}s"; // works
?>

```

Similarly, you can also have an array index or an object property parsed. With array indices, the closing square bracket () marks the end of the index. For object properties the same rules apply as to simple variables, though with object properties there doesn't exist a trick like the one with variables.

```

<?php
// These examples are specific to using arrays inside of strings.
// When outside of a string, always quote your array string keys
// and do not use {braces} when outside of strings either.

```

```

// Let's show all errors
error_reporting(E_ALL);

$fruits = array('strawberry' => 'red', 'banana' => 'yellow');

// Works but note that this works differently outside string-quotes
echo "A banana is $fruits[banana].";

// Works
echo "A banana is {$fruits['banana']}.";

// Works but PHP looks for a constant named banana first
// as described below.
echo "A banana is {$fruits[banana]}.";

// Won't work, use braces. This results in a parse error.
echo "A banana is $fruits['banana'].";

// Works
echo "A banana is " . $fruits['banana'] . ".";

// Works
echo "This square is $square->width meters broad.';

// Won't work. For a solution, see the complex syntax.
echo "This square is $square->width00 centimeters broad.";
?>

```

For anything more complex, you should use the complex syntax.

## Complex (curly) syntax

This isn't called complex because the syntax is complex, but because you can include complex expressions this way.

In fact, you can include any value that is in the namespace in strings with this syntax. You simply write the expression the same way as you would outside the string, and then include it in { and }. Since you can't escape '{', this syntax will only be recognised when the \$ is immediately following the {. (Use "{\\$" to get a literal "{\$"). Some examples to make it clear:

```

<?php
// Let's show all errors
error_reporting(E_ALL);

$great = 'fantastic';

// Won't work, outputs: This is { fantastic}
echo "This is { $great }";

// Works, outputs: This is fantastic
echo "This is {$great}";
echo "This is ${great}";

// Works

```

```

echo "This square is {$square->width}00 centimeters broad.";

// Works
echo "This works: {$arr[4][3]}";

// This is wrong for the same reason as $foo[bar] is wrong
// outside a string. In other words, it will still work but
// because PHP first looks for a constant named foo, it will
// throw an error of level E_NOTICE (undefined constant).
echo "This is wrong: {$arr[foo][3]}";

// Works. When using multi-dimensional arrays, always use
// braces around arrays when inside of strings
echo "This works: {$arr['foo'][3]}";

// Works.
echo "This works: " . $arr['foo'][3];

echo "You can even write {$obj->values[3]->name}";

echo "This is the value of the var named $name: {${$name}}";
?>

```

## String access and modification by character

Characters within strings may be accessed and modified by specifying the zero-based offset of the desired character after the string using square array-brackets like `$str[42]` so think of a string as an array of characters.

Note: They may also be accessed using braces like `$str{42}` for the same purpose. However, using square array-brackets is preferred because the {braces} style is deprecated as of PHP 6.

### Example 3. Some string examples

```

<?php
// Get the first character of a string
$str = 'This is a test.';
$first = $str[0];

// Get the third character of a string
$third = $str[2];

// Get the last character of a string.
$str = 'This is still a test.';
$last = $str[strlen($str)-1];

// Modify the last character of a string
$str = 'Look at the sea';
$str[strlen($str)-1] = 'e';

// Alternative method using {} is deprecated as of PHP 6
$third = $str{2};

?>

```

## Useful functions and operators

Strings may be concatenated using the '.' (dot) operator. Note that the '+' (addition) operator will not work for this. Please see String operators for more information.

There are a lot of useful functions for string modification.

There are also functions for URL-strings, and functions to encrypt/decrypt strings (mcrypt and mhash).

### Converting to string

You can convert a value to a string using the *(string)* cast, or the **strval()** function. String conversion is automatically done in the scope of an expression for you where a string is needed. This happens when you use the **echo()** or **print()** functions, or when you compare a variable value to a string.

A **boolean** **TRUE** value is converted to the string "1", the **FALSE** value is represented as "" (empty string). This way you can convert back and forth between boolean and string values.

An **integer** or a floating point number (**float**) is converted to a string representing the number with its digits (including the exponent part for floating point numbers).

Arrays are always converted to the string "Array", so you cannot dump out the contents of an **array** with **echo()** or **print()** to see what is inside them. To view one element, you'd do something like `echo $arr['foo']`. See below for tips on dumping/viewing the entire contents.

Objects are always converted to the string "Object". If you would like to print out the member variable values of an object for debugging reasons, read the paragraphs below. If you would like to find out the class name of which an object is an instance of, use **get\_class()**. As of PHP 5, **\_\_toString()** method is used if applicable.

Resources are always converted to strings with the structure "Resource id #1" where 1 is the unique number of the **resource** assigned by PHP during runtime. If you would like to get the type of the resource, use **get\_resource\_type()**.

**NULL** is always converted to an empty string.

As you can see above, printing out the arrays, objects or resources does not provide you any useful information about the values themselves. Look at the functions **print\_r()** and **var\_dump()** for better ways to print out values for debugging.

You can also convert PHP values to strings to store them permanently. This method is called serialization, and can be done with the function **serialize()**. You can also serialize PHP values to XML structures, if you have WDDX support in your PHP setup.

### String conversion to numbers

When a string is evaluated as a numeric value, the resulting value and type are determined as follows.

The string will evaluate as a **float** if it contains any of the characters '.', 'e', or 'E'. Otherwise, it will evaluate as an integer.

The value is given by the initial portion of the string. If the string starts with valid numeric data, this will be the value used. Otherwise, the value will be 0 (zero). Valid numeric data is an optional sign, followed by one or more digits (optionally containing a decimal point), followed by an optional exponent. The exponent is an 'e' or 'E' followed by one or more digits.

```
<?php
$foo = 1 + "10.5";                                // $foo is float (11.5)
$foo = 1 + "-1.3e3";                               // $foo is float (-1299)
$foo = 1 + "bob-1.3e3";                            // $foo is integer (1)
$foo = 1 + "bob3";                                 // $foo is integer (1)
$foo = 1 + "10 Small Pigs";                         // $foo is integer (11)
$foo = 4 + "10.2 Little Piggies"; // $foo is float (14.2)
$foo = "10.0 pigs " + 1;                           // $foo is float (11)
$foo = "10.0 pigs " + 1.0;                          // $foo is float (11)
?>
```

If you would like to test any of the examples in this section, you can cut and paste the examples and insert the following line to see for yourself what's going on:

```
<?php
echo "\$foo==\$foo; type is " . gettype ($foo) . "<br"
?>\n";
?>
```

Do not expect to get the code of one character by converting it to integer (as you would do in C for example). Use the functions **ord()** and **chr()** to convert between charcodes and characters.

## Arrays

An array in PHP is actually an ordered map. A map is a type that maps values to keys. This type is optimized in several ways, so you can use it as a real array, or a list (vector), hashtable (which is an implementation of a map), dictionary, collection, stack, queue and probably more. Because you can have another PHP array as a value, you can also quite easily simulate trees.

### Syntax

An **array** can be created by the **array()** language-construct. It takes a certain number of comma-separated *key => value* pairs.

- `array( [key =>] value`
- `,` ...
- `)`
- `// key may be an integer or string`
- `// value may be any value`
- `<?php`
- `$arr = array("foo" => "bar", 12 => true);`

- echo \$arr["foo"]; // bar
- echo \$arr[12]; // 1
- ?>

A key may be either an *integer* or a string. If a key is the standard representation of an integer, it will be interpreted as such (i.e. "8" will be interpreted as 8, while "08" will be interpreted as 08). Floats in key are truncated to integer. There are no different indexed and associative array types in PHP; there is only one array type, which can both contain integer and string indices.

A value can be of any PHP type.

```
<?php
$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));

echo $arr["somearray"][6]; // 5
echo $arr["somearray"][13]; // 9
echo $arr["somearray"]["a"]; // 42
?>
```

If you do not specify a key for a given value, then the maximum of the integer indices is taken, and the new key will be that maximum value + 1. If you specify a key that already has a value assigned to it, that value will be overwritten.

```
<?php
// This array is the same as ...
array(5 => 43, 32, 56, "b" => 12);

// ...this array
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

## Creating/modifying with square-bracket syntax

You can also modify an existing array by explicitly setting values in it.

This is done by assigning values to the array while specifying the key in brackets. You can also omit the key, add an empty pair of brackets ("[]") to the variable name in that case.

```
$arr[key] = value;
$arr[] = value;
// key may be an integer or string
// value may be any value
```

If \$arr doesn't exist yet, it will be created. So this is also an alternative way to specify an array. To change a certain value, just assign a new value to an element specified with its key. If you want to remove a key/value pair, you need to **unset()** it.

```
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56; // This is the same as $arr[13] = 56;
             // at this point of the script

$arr["x"] = 42; // This adds a new element to
```

```

        // the array with key "x"

unset($arr[5]); // This removes the element from the array

unset($arr);    // This deletes the whole array
?>

```

## Objects

### Object Initialization

To initialize an object, you use the `new` statement to instantiate the object to a variable.

```

<?php
class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
?>

```

### Converting to object

If an object is converted to an object, it is not modified. If a value of any other type is converted to an object, a new instance of the `stdClass` built in class is created. If the value was `NULL`, the new instance will be empty. Array converts to an object with properties named by array keys and with corresponding values. For any other value, a member variable named `scalar` will contain the value.

```

<?php
$obj = (object) 'ciao';
echo $obj->scalar; // outputs 'ciao'
?>

```

## NULL

The special `NULL` value represents that a variable has no value. `NULL` is the only possible value of type **NULL**.

Note: The null type was introduced in PHP 4.

A variable is considered to be `NULL` if

- it has been assigned the constant `NULL`.
- it has not been set to any value yet.
- it has been `unset()`.

## Syntax

There is only one value of type **NULL**, and that is the case-insensitive keyword **NULL**.

```
<?php  
$var = NULL;  
?>
```

See also **is\_null()** and **unset()**.

## Variables

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.

Variable names follow the same rules as other labels in PHP. A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus: '[a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\*'

Note: For our purposes here, a letter is a-z, A-Z, and the ASCII characters from 127 through 255 (0x7f-0xff). In PHP 3, variables are always assigned by value. That is to say, when you assign an expression to a variable, the entire value of the original expression is copied into the destination variable. This means, for instance, that after assigning one variable's value to another, changing one of those variables will have no effect on the other. For more information on this kind of assignment, see the chapter on Expressions.

As of PHP 4, PHP offers another way to assign values to variables: assign by reference. This means that the new variable simply references (in other words, "becomes an alias for" or "points to") the original variable. Changes to the new variable affect the original, and vice versa.

To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable).

## Predefined variables

PHP provides a large number of predefined variables to any script which it runs. Many of these variables, however, cannot be fully documented as they are dependent upon which server is running, the version and setup of the server, and other factors. Some of these variables will not be available when PHP is run on the command line.

## Variable scope

The scope of a variable is the context within which it is defined. For the most part all PHP variables only have a single scope. This single scope spans included and required files as well. For example:

```
<?php  
$a = 1;  
include 'b.inc';  
?>
```

Here the `$a` variable will be available within the included `b.inc` script. However, within user-defined functions a local function scope is introduced. Any variable used inside a function is by default limited to the local function scope. For example:

```

<?php
$a = 1; /* global scope */

function Test()
{
    echo $a; /* reference to local scope variable */
}

Test();
?>

```

This script will not produce any output because the echo statement refers to a local version of the \$a variable, and it has not been assigned a value within this scope. You may notice that this is a little bit different from the C language in that global variables in C are automatically available to functions unless specifically overridden by a local definition. This can cause some problems in that people may inadvertently change a global variable. In PHP global variables must be declared global inside a function if they are going to be used in that function.

## Using static variables

Another important feature of variable scoping is the static variable. A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope.

### Example use of static variables

```

<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>

```

Now, every time the Test() function is called it will print the value of \$a and increment it.

Static variables also provide one way to deal with recursive functions. A recursive function is one which calls itself. Care must be taken when writing a recursive function because it is possible to make it recurse indefinitely. You must make sure you have an adequate way of terminating the recursion. The following simple function recursively counts to 10, using the static variable \$count to know when to stop:

## References with global and static variables

The Zend Engine 1, driving PHP 4, implements the static and global modifier for variables in terms of references. For example, a true global variable imported inside a function scope with the *global* statement actually creates a reference to the global variable. This can lead to unexpected behaviour which the following example addresses:

```

<?php
function test_global_ref() {
    global $obj;
    $obj = &new stdclass;
}

```

```

function test_global_noref() {
    global $obj;
    $obj = new stdclass;
}

test_global_ref();
var_dump($obj);
test_global_noref();
var_dump($obj);
?>

```

Executing this example will result in the following output:

```

NULL
object(stdClass)(0) {
}

```

## Variable variables

Sometimes it is convenient to be able to have variable variable names. That is, a variable name which can be set and used dynamically. A normal variable is set with a statement such as:

```

<?php
$a = 'hello';
?>

```

A variable variable takes the value of a variable and treats that as the name of a variable. In the above example, `hello`, can be used as the name of a variable by using two dollar signs. i.e.

```

<?php
$$a = 'world';
?>

```

At this point two variables have been defined and stored in the PHP symbol tree: `$a` with contents "hello" and `$hello` with contents "world". Therefore, this statement:

```

<?php
echo "$a ${$a}";
?>

```

produces the exact same output as:

```

<?php
echo "$a $hello";
?>

```

i.e. they both produce: hello world.

# Constants

A constant is an identifier (name) for a simple value. As the name suggests, that value cannot change during the execution of the script (except for magic constants, which aren't actually constants). A constant is case-sensitive by default. By convention, constant identifiers are always uppercase.

The name of a constant follows the same rules as any label in PHP. A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thusly: `[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*`

## Syntax

You can define a constant by using the **define()**-function. Once a constant is defined, it can never be changed or undefined.

Only scalar data (**Boolean**, **integer**, **float** and **string**) can be contained in constants. Do not define **resource** constants.

You can get the value of a constant by simply specifying its name. Unlike with variables, you should not prepend a constant with a \$. You can also use the function **constant()** to read a constant's value if you wish to obtain the constant's name dynamically. Use **get\_defined\_constants()** to get a list of all defined constants.

Note: Constants and (global) variables are in a different namespace. This implies that for example **TRUE** and **\$TRUE** are generally different.

If you use an undefined constant, PHP assumes that you mean the name of the constant itself, just as if you called it as a **string** (CONSTANT vs "CONSTANT"). An error of level **E\_NOTICE** will be issued when this happens. See also the manual entry on why **\$foo[bar]** is wrong (unless you first define() bar as a constant). If you simply want to check if a constant is set, use the **defined()** function.

These are the differences between constants and variables:

- Constants do not have a dollar sign (\$) before them;
- Constants may only be defined using the **define()** function, not by simple assignment;
- Constants may be defined and accessed anywhere without regard to variable scoping rules;
- Constants may not be redefined or undefined once they have been set; and
- Constants may only evaluate to scalar values.

### Example. Defining Constants

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.
?>
```

# Expressions

- Expressions are the most important building stones of PHP. In PHP, almost anything you write is an expression. The simplest yet most accurate way to define an expression is "anything that has a value".

The most basic forms of expressions are constants and variables. When you type "\$a = 5", you're assigning '5' into \$a. '5', obviously, has the value 5, or in other words '5' is an expression with the value of 5 (in this case, '5' is an integer constant).

After this assignment, you'd expect \$a's value to be 5 as well, so if you wrote \$b = \$a, you'd expect it to behave just as if you wrote \$b = 5. In other words, \$a is an expression with the value of 5 as well. If everything works right, this is exactly what will happen.

Of course, values in PHP don't have to be integers, and very often they aren't. PHP supports four scalar value types: integer values, floating point values (float), string values and boolean values (scalar values are values that you can't 'break' into smaller pieces, unlike arrays, for instance). PHP also supports two composite (non-scalar) types: arrays and objects. Each of these value types can be assigned into variables or returned from functions.

PHP takes expressions much further, in the same way many other languages do. PHP is an expression-oriented language, in the sense that almost everything is an expression. Consider the example we've already dealt with, '\$a = 5'. It's easy to see that there are two values involved here, the value of the integer constant '5', and the value of \$a which is being updated to 5 as well. But the truth is that there's one additional value involved here, and that's the value of the assignment itself. The assignment itself evaluates to the assigned value, in this case 5. In practice, it means that '\$a = 5', regardless of what it does, is an expression with the value 5. Thus, writing something like '\$b = (\$a = 5)' is like writing '\$a = 5; \$b = 5;' (a semicolon marks the end of a statement). Since assignments are parsed in a right to left order, you can also write '\$b = \$a = 5'.

Another good example of expression orientation is pre- and post-increment and decrement. Users of PHP and many other languages may be familiar with the notation of variable++ and variable--. These are increment and decrement operators. In PHP/FI 2, the statement '\$a++' has no value (is not an expression), and thus you can't assign it or use it in any way. PHP enhances the increment/decrement capabilities by making these expressions as well, like in C. In PHP, like in C, there are two types of increment - pre-increment and post-increment. Both pre-increment and post-increment essentially increment the variable, and the effect on the variable is identical. The difference is with the value of the increment expression. Pre-increment, which is written '++\$variable', evaluates to the incremented value (PHP increments the variable before reading its value, thus the name 'pre-increment'). Post-increment, which is written '\$variable++' evaluates to the original value of \$variable, before it was incremented (PHP increments the variable after reading its value, thus the name 'post-increment').

A very common type of expressions are comparison expressions. These expressions evaluate to either FALSE or TRUE. PHP supports > (bigger than), >= (bigger than or equal to), == (equal), != (not equal), < (smaller than) and <= (smaller than or equal to). The language also supports a set of strict equivalence operators: === (equal to and same type) and !== (not equal to or not same type). These expressions are most commonly used inside conditional execution, such as if statements.

## Operators

An operator is something that you feed with one or more values (or expressions, in programming jargon) which yields another value (so that the construction itself becomes an expression). So you can think of functions or constructions that return a value (like print) as operators and those that return nothing (like echo) as any other thing.

There are three types of operators. Firstly there is the unary operator which operates on only one value, for example ! (the negation operator) or ++ (the increment operator). The second group are termed binary operators; this group contains most of the operators that PHP supports, and a list follows below in the section **Operator Precedence**.

The third group is the ternary operator: `?:`. It should be used to select between two expressions depending on a third one, rather than to select two sentences or paths of execution. Surrounding ternary expressions with parentheses is a very good idea.

## Operator Precedence

The precedence of an operator specifies how "tightly" it binds two expressions together. For example, in the expression `1 + 5 * 3`, the answer is 16 and not 18 because the multiplication ("`*`") operator has a higher precedence than the addition ("`+`") operator. Parentheses may be used to force precedence, if necessary. For instance: `(1 + 5) * 3` evaluates to 18. If operator precedence is equal, left to right associativity is used.

The following table lists the precedence of operators with the highest-precedence operators listed at the top of the table. Operators on the same line have equal precedence, in which case their associativity decides which order to evaluate them in.

**Table 2. Operator Precedence**

Associativity	Operators	Additional Information
non-associative	<code>new</code>	<u>new</u>
left	<code>[</code>	<u>array()</u>
non-associative	<code>++ --</code>	<u>increment/decrement</u>
non-associative	<code>~ - (int) (float) (string) (array) (object) @</code>	<u>types</u>
non-associative	<code>instanceof</code>	<u>types</u>
right	<code>!</code>	<u>logical</u>
left	<code>* / %</code>	<u>arithmetic</u>
left	<code>+ - .</code>	<u>arithmetic and string</u>
left	<code>&lt;&lt; &gt;&gt;</code>	<u>bitwise</u>
non-associative	<code>&lt; &lt;= &gt; &gt;=</code>	<u>comparison</u>
non-associative	<code>== != === !==</code>	<u>comparison</u>
left	<code>&amp;</code>	<u>bitwise and references</u>
left	<code>^</code>	<u>bitwise</u>
left	<code> </code>	<u>bitwise</u>
left	<code>&amp;&amp;</code>	<u>logical</u>
left	<code>  </code>	<u>logical</u>
left	<code>? :</code>	<u>ternary</u>
right	<code>= += -= *= /= .= %= &amp;=  = ^= &lt;&lt;= &gt;&gt;=</code>	<u>assignment</u>
left	<code>and</code>	<u>logical</u>
left	<code>xor</code>	<u>logical</u>
left	<code>or</code>	<u>logical</u>
left	<code>,</code>	<u>many uses</u>

Left associativity means that the expression is evaluated from left to right, right associativity means the opposite.

## Arithmetic Operators

Remember basic arithmetic from school? These work just like those.

Table . Arithmetic Operators

Example	Name	Result
<code>-\$a</code>	Negation	Opposite of \$a.
<code>\$a + \$b</code>	Addition	Sum of \$a and \$b.
<code>\$a - \$b</code>	Subtraction	Difference of \$a and \$b.
<code>\$a * \$b</code>	Multiplication	Product of \$a and \$b.
<code>\$a / \$b</code>	Division	Quotient of \$a and \$b.
<code>\$a % \$b</code>	Modulus	Remainder of \$a divided by \$b.

The division operator ("/") returns a float value anytime, even if the two operands are integers (or strings that get converted to integers).

## Assignment Operators

The basic assignment operator is "`=`". Your first inclination might be to think of this as "equal to". Don't. It really means that the left operand gets set to the value of the expression on the right (that is, "gets set to").

The value of an assignment expression is the value assigned. That is, the value of "`$a = 3`" is 3. This allows you to do some tricky things:

```
<?php
$a = ($b = 4) + 5; // $a is equal to 9 now, and $b has been set to
4.
?>
```

## Bitwise Operators

Bitwise operators allow you to turn specific bits within an integer on or off. If both the left- and right-hand parameters are strings, the bitwise operator will operate on the characters' ASCII values.

```
<?php
echo 12 ^ 9; // Outputs '5'
echo "12" ^ "9"; // Outputs the Backspace character (ascii 8)
                  // ('1' (ascii 49)) ^ ('9' (ascii 57)) = #8
echo "hallo" ^ "hello"; // Outputs the ascii values #0 #4 #0 #0 #0
                        // 'a' ^ 'e' = #4
?>
```

Table 3. Bitwise Operators

Example	Name	Result
<code>\$a &amp; \$b</code>	And	Bits that are set in both \$a and \$b are set.
<code>\$a   \$b</code>	Or	Bits that are set in either \$a or \$b are set.

Example	Name	Result
<code>\$a ^ \$b</code>	Xor	Bits that are set in \$a or \$b but not both are set.
<code>~ \$a</code>	Not	Bits that are set in \$a are not set, and vice versa.
<code>\$a &lt;&lt; \$b</code>	Shift left	Shift the bits of \$a \$b steps to the left (each step means "multiply by two")
<code>\$a &gt;&gt; \$b</code>	Shift right	Shift the bits of \$a \$b steps to the right (each step means "divide by two")

## Comparison Operators

Comparison operators, as their name implies, allow you to compare two values. **Comparison Operators**

Example	Name	Result
<code>\$a == \$b</code>	Equal	<b>TRUE</b> if \$a is equal to \$b.
<code>\$a === \$b</code>	Identical	<b>TRUE</b> if \$a is equal to \$b, and they are of the same type. (introduced in PHP 4)
<code>\$a != \$b</code>	Not equal	<b>TRUE</b> if \$a is not equal to \$b.
<code>\$a &lt;&gt; \$b</code>	Not equal	<b>TRUE</b> if \$a is not equal to \$b.
<code>\$a !== \$b</code>	Not identical	<b>TRUE</b> if \$a is not equal to \$b, or they are not of the same type. (introduced in PHP 4)
<code>\$a &lt; \$b</code>	Less than	<b>TRUE</b> if \$a is strictly less than \$b.
<code>\$a &gt; \$b</code>	Greater than	<b>TRUE</b> if \$a is strictly greater than \$b.
<code>\$a &lt;= \$b</code>	Less than or equal to	<b>TRUE</b> if \$a is less than or equal to \$b.
<code>\$a &gt;= \$b</code>	Greater than or equal to	<b>TRUE</b> if \$a is greater than or equal to \$b.

## Ternary Operator

Another conditional operator is the ":" (or ternary) operator.

The expression `(expr1) ? (expr2) : (expr3)` evaluates to `expr2` if `expr1` evaluates to TRUE, and `expr3` if `expr1` evaluates to FALSE.

## Error Control Operators

PHP supports one error control operator: the at sign (@). When prepended to an expression in PHP, any error messages that might be generated by that expression will be ignored.

## Execution Operators

PHP supports one execution operator: backticks (` `). Note that these are not single-quotes! PHP will attempt to execute the contents of the backticks as a shell command; the output will be returned (i.e., it won't simply be dumped to output; it can be assigned to a variable). Use of the backtick operator is identical to **shell\_exec()**.

## Incrementing/Decrementing Operators

PHP supports C-style pre- and post-increment and decrement operators.

Note: The increment/decrement operators do not affect boolean values. Decrementing NULL values has no effect too, but incrementing them results in 1.

**Table 15-6. Increment/decrement Operators**

Example	Name	Effect
<code>++\$a</code>	Pre-increment	Increments \$a by one, then returns \$a.
<code>\$a++</code>	Post-increment	Returns \$a, then increments \$a by one.
<code>--\$a</code>	Pre-decrement	Decrements \$a by one, then returns \$a.
<code>\$a--</code>	Post-decrement	Returns \$a, then decrements \$a by one.

## Logical Operators

**Table 4. Logical Operators**

Example	Name	Result
<code>\$a and \$b</code>	And	<b>TRUE</b> if both \$a and \$b are <b>TRUE</b> .
<code>\$a or \$b</code>	Or	<b>TRUE</b> if either \$a or \$b is <b>TRUE</b> .
<code>\$a xor \$b</code>	Xor	<b>TRUE</b> if either \$a or \$b is <b>TRUE</b> , but not both.
<code>! \$a</code>	Not	<b>TRUE</b> if \$a is not <b>TRUE</b> .
<code>\$a &amp;&amp; \$b</code>	And	<b>TRUE</b> if both \$a and \$b are <b>TRUE</b> .
<code>\$a    \$b</code>	Or	<b>TRUE</b> if either \$a or \$b is <b>TRUE</b> .

## String Operators

There are two **string** operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side.

## Array Operators

**Table 5. Array Operators**

Example	Name	Result
<code>\$a + \$b</code>	Union	Union of \$a and \$b.
<code>\$a == \$b</code>	Equality	<code>TRUE</code> if \$a and \$b have the same key/value pairs.
<code>\$a === \$b</code>	Identity	<code>TRUE</code> if \$a and \$b have the same key/value pairs in the same order and of the same types.
<code>\$a != \$b</code>	Inequality	<code>TRUE</code> if \$a is not equal to \$b.
<code>\$a &lt;&gt; \$b</code>	Inequality	<code>TRUE</code> if \$a is not equal to \$b.
<code>\$a !== \$b</code>	Non-identity	<code>TRUE</code> if \$a is not identical to \$b.

## Control Structures

Any PHP script is built out of a series of statements. A statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing (an empty statement). Statements usually end with a semicolon. In addition, statements can be grouped into a statement-group by encapsulating a group of statements with curly braces. A statement-group is a statement by itself as well. The various statement types are described in this section.

### if

The *if* construct is one of the most important features of many languages, PHP included. It allows for conditional execution of code fragments. PHP features an *if* structure that is similar to that of C:

```
if (expr)
    statement
```

As described in the section about expressions, **expression** is evaluated to its Boolean value. If **expression** evaluates to `TRUE`, PHP will execute *statement*, and if it evaluates to `FALSE` - it'll ignore it..

The following example would display a is bigger than b if `$a` is bigger than `$b`:

```
<?php
if ($a > $b)
    echo "a is bigger than b";
?>
```

Often you'd want to have more than one statement to be executed conditionally. Of course, there's no need to wrap each statement with an *if* clause. Instead, you can group several statements into a statement group. For example, this code would display a is bigger than b if `$a` is bigger than `$b`, and would then assign the value of `$a` into `$b`:

```
<?php
if ($a > $b) {
    echo "a is bigger than b";
    $b = $a;
}
?>
```

If statements can be nested indefinitely within other if statements, which provides you with complete flexibility for conditional execution of the various parts of your program.

## else

Often you'd want to execute a statement if a certain condition is met, and a different statement if the condition is not met. This is what *else* is for. *else* extends an *if* statement to execute a statement in case the expression in the *if* statement evaluates to FALSE. For example, the following code would display a is bigger than b if *\$a* is bigger than *\$b*, and a is NOT bigger than b otherwise:

```
<?php
if ($a > $b) {
    echo "a is bigger than b";
} else {
    echo "a is NOT bigger than b";
}
?>
```

The *else* statement is only executed if the *if* expression evaluated to FALSE, and if there were any *elseif* expressions - only if they evaluated to FALSE as well (see *elseif*).

## elseif

*elseif*, as its name suggests, is a combination of *if* and *else*. Like *else*, it extends an *if* statement to execute a different statement in case the original *if* expression evaluates to FALSE. However, unlike *else*, it will execute that alternative expression only if the *elseif* conditional expression evaluates to TRUE. For example, the following code would display a is bigger than b, a equal to b or a is smaller than b:

```
<?php
if ($a > $b) {
    echo "a is bigger than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is smaller than b";
}
?>
```

There may be several *elseif*s within the same *if* statement. The first *elseif* expression (if any) that evaluates to TRUE would be executed. In PHP, you can also write 'else if' (in two words) and the behavior would be identical to the one of 'elseif' (in a single word). The syntactic meaning is slightly different (if you're familiar with C, this is the same behavior) but the bottom line is that both would result in exactly the same behavior.

The *elseif* statement is only executed if the preceding *if* expression and any preceding *elseif* expressions evaluated to FALSE, and the current *elseif* expression evaluated to TRUE.

## while

*while* loops are the simplest type of loop in PHP. They behave just like their C counterparts. The basic form of a *while* statement is:

```
while (expr)
    statement
```

The meaning of a *while* statement is simple. It tells PHP to execute the nested statement(s) repeatedly, as long as the *while* expression evaluates to **TRUE**. The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time PHP runs the statements in the loop is one iteration). Sometimes, if the *while* expression evaluates to **FALSE** from the very beginning, the nested statement(s) won't even be run once.

Like with the *if* statement, you can group multiple statements within the same *while* loop by surrounding a group of statements with curly braces, or by using the alternate syntax:

```
while (expr):
    statement
    ...
endwhile;
```

The following examples are identical, and both print numbers from 1 to 10:

```
<?php
/* example 1 */

$i = 1;
while ($i <= 10) {
    echo $i++; /* the printed value would be
                  $i before the increment
                  (post-increment) */
}

/* example 2 */

$i = 1;
while ($i <= 10):
    echo $i;
    $i++;
endwhile;
?>
```

## do-while

*do-while* loops are very similar to *while* loops, except the truth expression is checked at the end of each iteration instead of in the beginning. The main difference from regular *while* loops is that the first iteration of a *do-while* loop is guaranteed to run (the truth expression is only checked at the end of the iteration), whereas it's may not necessarily run with a regular *while* loop (the truth expression is checked at the beginning of each iteration, if it evaluates to **FALSE** right from the beginning, the loop execution would end immediately).

There is just one syntax for do-while loops:

```
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```

The above loop would run one time exactly, since after the first iteration, when truth expression is checked, it evaluates to **FALSE** ( $\$i$  is not bigger than 0) and the loop execution ends.

Advanced C users may be familiar with a different usage of the *do-while* loop, to allow stopping execution in the middle of code blocks, by encapsulating them with *do-while* (0), and using the **break** statement. The following code fragment demonstrates this:

```
<?php
do {
    if ($i < 5) {
        echo "i is not big enough";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    echo "i is ok";
    /* process i */
} while (0);
?>
```

Don't worry if you don't understand this right away or at all. You can code scripts and even powerful scripts without using this 'feature'.

## for

*for* loops are the most complex loops in PHP. They behave like their C counterparts. The syntax of a *for* loop is:

```
for (expr1; expr2; expr3)
    statement
```

The first expression (*expr1*) is evaluated (executed) once unconditionally at the beginning of the loop.

In the beginning of each iteration, *expr2* is evaluated. If it evaluates to **TRUE**, the loop continues and the nested statement(s) are executed. If it evaluates to **FALSE**, the execution of the loop ends.

At the end of each iteration, *expr3* is evaluated (executed).

Each of the expressions can be empty or contain multiple expressions separated by commas. Comma separated expressions in *expr2* are treated similarly to being separated by the **||**

operator but has a lower precedence than `||`. `expr2` being empty means the loop `should` be run indefinitely (PHP implicitly considers it as `TRUE`, like C). This may not be as useless as you might think, since often you'd want to end the loop using a conditional `break` statement instead of using the `for` truth expression.

Consider the following examples. All of them display numbers from 1 to 10:

```
<?php
/* example 1 */

for ($i = 1; $i <= 10; $i++) {
    echo $i;
}

/* example 2 */

for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}

/* example 3 */

$i = 1;
for (; ; ) {
    if ($i > 10) {
        break;
    }
    echo $i;
    $i++;
}

/* example 4 */

for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
?>
```

Of course, the first example appears to be the nicest one (or perhaps the fourth), but you may find that being able to use empty expressions in `for` loops comes in handy in many occasions.

PHP also supports the alternate "colon syntax" for `for` loops.

```
for (expr1; expr2; expr3):
    statement
    ...
endfor;
```

## foreach

PHP 4 introduced a `foreach` construct, much like Perl and some other languages. This simply gives an easy way to iterate over arrays. `foreach` works only on arrays, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable. There are two syntaxes; the second is a minor but useful extension of the first:

```

foreach (array_expression as $value)
    statement
foreach (array_expression as $key => $value)
    statement

```

The first form loops over the array given by *array\_expression*. On each loop, the value of the current element is assigned to *\$value* and the internal array pointer is advanced by one (so on the next loop, you'll be looking at the next element).

The second form does the same thing, except that the current element's key will be assigned to the variable *\$key* on each loop.

## **break**

*break* ends execution of the current *for*, *foreach*, *while*, *do-while* or *switch* structure.

*break* accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.

```

<?php
$arr = array('one', 'two', 'three', 'four', 'stop', 'five');
while (list(), $val) = each($arr)) {
    if ($val == 'stop') {
        break; /* You could also write 'break 1;' here. */
    }
    echo "$val<br />\n";
}

/* Using the optional argument. */

$i = 0;
while (++$i) {
    switch ($i) {
        case 5:
            echo "At 5<br />\n";
            break 1; /* Exit only the switch. */
        case 10:
            echo "At 10; quitting<br />\n";
            break 2; /* Exit the switch and the while. */
        default:
            break;
    }
}
?>

```

## **continue**

*continue* is used within looping structures to skip the rest of the current loop iteration and continue execution at the condition evaluation and then the beginning of the next iteration.

Note: Note that in PHP the switch statement is considered a looping structure for the purposes of *continue*.

*continue* accepts an optional numeric argument which tells it how many levels of enclosing loops it should skip to the end of.

```
<?php
```

```

while (list($key, $value) = each($arr)) {
    if (!$key % 2) { // skip odd members
        continue;
    }
    do_something_odd($value);
}

$i = 0;
while ($i++ < 5) {
    echo "Outer<br />\n";
    while (1) {
        echo "&nbsp;&nbsp;Middle<br />\n";
        while (1) {
            echo "&nbsp;&nbsp;Inner<br />\n";
            continue 3;
        }
        echo "This never gets output.<br />\n";
    }
    echo "Neither does this.<br />\n";
}
?>

```

Omitting the semicolon after *continue* can lead to confusion. Here's an example of what you shouldn't do.

```

<?php
for ($i = 0; $i < 5; ++$i) {
    if ($i == 2)
        continue
    print "$i\n";
}
?>

```

One can expect the result to be :

```

0
1
3
4

```

but this script will output :

```

2

```

because the return value of the `print()` call is `int(1)`, and it will look like the optional numeric argument mentioned above.

## **switch**

The `switch` statement is similar to a series of `IF` statements on the same expression. In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the `switch` statement is for.

## **declare**

The `declare` construct is used to set execution directives for a block of code. The syntax of `declare` is similar to the syntax of other flow control constructs:

```
declare (directive)
statement
```

The `directive` section allows the behavior of the `declare` block to be set. Currently only one directive is recognized: the `ticks` directive. (See below for more information on the **ticks'** directive)

The **statement** part of the **declare** block will be executed -- how it is executed and what side effects occur during execution may depend on the directive set in the `directive` block.

## Ticks

A tick is an event that occurs for every *N* low-level statements executed by the parser within the `declare` block. The value for *N* is specified using `ticks=N` within the `declare` block's `directive` section.

The event(s) that occur on each tick are specified using the **register\_tick\_function()**. See the example below for more details. Note that more than one event can occur for each tick.

## return

If called from within a function, the `return()` statement immediately ends execution of the current function, and returns its argument as the value of the function call. `return()` will also end the execution of an `eval()` statement or script file.

If called from the global scope, then execution of the current script file is ended. If the current script file was `include()`, `ed` or `return()`ed, then control is passed back to the calling file. Furthermore, if the current script file was `include()`, `ed`, then the value given to `return()` will be returned as the value of the `include()`, `call`. If `return()` is called from within the main script file, then script execution ends. If the current script file was named by the `auto-prepend_file` or `auto_append_file` configuration options in `php.ini`, then that script file's execution is ended.

## require()

The `require()` statement includes and evaluates the specific file.

`require()` includes and evaluates a specific file. Detailed information on how this inclusion works is described in the documentation for `include()`.

`require()` and `include()` are identical in every way except how they handle failure. `include()` produces a Warning while `require()` results in a Fatal Error. In other words, don't hesitate to use `require()` if you want a missing file to halt processing of the page. `include()` does not behave this way, the script will continue regardless. Be sure to have an appropriate `include_path` setting as well.

## include()

The include() statement includes and evaluates the specified file.

The documentation below also applies to require(). The two constructs are identical in every way except how they handle failure. include() produces a Warning while require() results in a Fatal Error. In other words, use require() if you want a missing file to halt processing of the page. include() does not behave this way, the script will continue regardless. Be sure to have an appropriate include\_path setting as well. Be warned that parse error in included file doesn't cause processing halting in PHP versions prior to PHP 4.3.5. Since this version, it does.

Files for including are first looked in include\_path relative to the current working directory and then in include\_path relative to the directory of current script. E.g. if your include\_path is ., current working directory is /www/, you included **include/a.php** and there is *include "b.php"* in that file, **b.php** is first looked in /WWW/ and then in /www/include/. If filename begins with ./ or ../, it is looked only in include\_path relative to the current working directory.

When a file is included, the code it contains inherits the variable scope of the line on which the include occurs. Any variables available at that line in the calling file will be available within the called file, from that point forward. However, all functions and classes defined in the included file have the global scope.

## Functions

### User-defined functions

A function may be defined using syntax such as the following:

Pseudo code to demonstrate function uses

```
<?php
function foo($arg_1, $arg_2, /* ... */ $arg_n)
{
    echo "Example function.\n";
    return $retval;
}
?>
```

Any valid PHP code may appear inside a function, even other functions and class definitions.

Function names follow the same rules as other labels in PHP. A valid function name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus: [a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\*.

### Function arguments

Information may be passed to functions via the argument list, which is a comma-delimited list of expressions.

PHP supports passing arguments by value (the default), passing by reference, and default argument values. Variable-length argument lists are supported only in PHP 4 and later; see Variable-length argument lists and the function references for func\_num\_args(), func\_get\_arg(), and func\_get\_args() for more information. A similar effect can be achieved in PHP 3 by passing an array of arguments to a function:

Passing arrays to functions

```
<?php
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
?>
```

## Returning values

Values are returned by using the optional return statement. Any type may be returned, including lists and objects. This causes the function to end its execution immediately and pass control back to the line from which it was called. See return() for more information.

## Variable functions

PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it. Among other things, this can be used to implement callbacks, function tables, and so forth.

Variable functions won't work with language constructs such as echo(), print(), unset(), iset(), empty(), include(), require() and the like. You need to use your own wrapper function to utilize any of these constructs as variable functions.

## Internal (built-in) functions

PHP comes standard with many functions and constructs. There are also functions that require specific PHP extensions compiled in otherwise you'll get fatal "undefined function" errors. For example, to use image functions such as imagecreatetruecolor(), you'll need your PHP compiled with GD support. Or, to use mysql\_connect() you'll need your PHP compiled in with MySQL support. There are many core functions that are included in every version of PHP like the string and variable functions. A call to phpinfo() or get\_loaded\_extensions() will show you which extensions are loaded into your PHP. Also note that many extensions are enabled by default and that the PHP manual is split up by extension. See the configuration, installation, and individual extension chapters, for information on how to setup your PHP.

It's important to realize what a function returns or if a function works directly on a passed in value. For example, str\_replace() will return the modified string while usort() works on the actual passed in variable itself. Each manual page also has specific information for each function like information on function parameters, behavior changes, return values for both success and failure, and availability information. Knowing these important (yet often subtle) differences is crucial for writing correct PHP code.

## Internal (built-in) functions

PHP comes standard with many functions and constructs. There are also functions that require specific PHP extensions compiled in otherwise you'll get fatal "undefined function" errors. For example, to use image functions such as imagecreatetruecolor(), you'll need your PHP compiled with GD support. Or, to use mysql\_connect() you'll need your PHP compiled in with MySQL support. There are many core functions that are included in every version of PHP like the string and variable functions. A call to phpinfo() or get\_loaded\_extensions() will show you which extensions are loaded into your PHP. Also note that many extensions are enabled by default and that the PHP manual is split up by extension.

It's important to realize what a function returns or if a function works directly on a passed in value. For example, str\_replace() will return the modified string while usort() works on the actual passed in variable itself. Each manual page also has specific information for each function like information on function parameters, behavior changes, return values for both success and failure,

and availability information. Knowing these important (yet often subtle) differences is crucial for writing correct PHP code.

## Exceptions

PHP 5 has an exception model similar to that of other programming languages. An exception can be thrown, and caught ("caught") within PHP. Code may be surrounded in a try block, to facilitate the catching of potential exceptions. Each try must have at least one corresponding catch block. Multiple catch blocks can be used to catch different classes of exceptions. Normal execution (when no exception is thrown within the try block, or when a catch matching the thrown exception's class is not present) will continue after that last catch block defined in sequence. Exceptions can be thrown (or re-thrown) within a catch block.

When an exception is thrown, code following the statement will not be executed, and PHP will attempt to find the first matching catch block. If an exception is not caught, a PHP Fatal Error will be issued with an "Uncaught Exception ..." message, unless a handler has been defined with set\_exception\_handler().

## HTTP authentication with PHP

The HTTP Authentication hooks in PHP are only available when it is running as an Apache module and is hence not available in the CGI version. In an Apache module PHP script, it is possible to use the header() function to send an "Authentication Required" message to the client browser causing it to pop up a Username/Password input window. Once the user has filled in a username and a password, the URL containing the PHP script will be called again with the predefined variables PHP\_AUTH\_USER, PHP\_AUTH\_PW, and AUTH\_TYPE set to the user name, password and authentication type respectively. These predefined variables are found in the \$\_SERVER and \$HTTP\_SERVER\_VARS arrays. Both "Basic" and "Digest" (since PHP 5.1.0) authentication methods are supported. See the header() function for more information.

Basic HTTP Authentication example

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="My Realm"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Text to send if user hits Cancel button';
    exit;
} else {
    echo "<p>Hello {$_SERVER['PHP_AUTH_USER']}.</p>";
    echo "<p>You entered {$_SERVER['PHP_AUTH_PW']} as your password.</p>";
}
?>
```

## Sessions

Session support in PHP consists of a way to preserve certain data across subsequent accesses. This enables you to build more customized applications and increase the appeal of your web site.

## ANNEXURE

## Installation and Configuration

### General Installation Considerations:

Before starting the installation, first you need to know what you want to use PHP for. There are three main fields you can use PHP,

- Websites and web applications (server-side scripting)
- Command line scripting
- Desktop (GUI) applications

For the first and most common form, you need three things: PHP itself, a web server and a web browser. You probably already have a web browser, and depending on your operating system setup, you may also have a web server (e.g. Apache on Linux and MacOS X; IIS on Windows). You may also rent web space at a company. This way, you don't need to set up anything on your own, only write your PHP scripts, upload it to the server you rent, and see the results in your browser.

### ***Debian GNU/Linux installation notes***

This section contains notes and hints specific to installing PHP on Debian GNU/Linux.

### Using APT

While you can just download the PHP source and compile it yourself, using Debian's packaging system is the simplest and cleanest method of installing PHP. If you are not familiar with building software on Linux, this is the way to go.

The first decision you need to make is whether you want to install Apache 1.3.x or Apache 2.x. The corresponding PHP packages are respectively named libapache-mod-php\* and libapache2-mod-php\*. The steps given below will use Apache 1.3.x. Please note that, as of this writing, there is no official Debian packages of PHP 5. Then the steps given below will install PHP 4.

PHP is available in Debian as CGI or CLI flavour too, named respectively php4-cgi and php4-cli. If you need them, you'll just have to reproduce the following steps with the good package names. Another special package you'd want to install is php4-pear. It contains a minimal PEAR installation and the pear commandline utility.

If you need more recent packages of PHP than the Debian's stable ones or if some PHP modules lacks the Debian official repository, perhaps you should take a look at <http://www.apt-get.org/>. One of the results found should be Dotdeb. This unofficial repository is maintained by Guillaume Plessis and contains Debian packages of the most recent versions of PHP 4 and PHP 5. To use it, just add the two following lines to your /etc/apt/sources.list and run apt-get update :

Example. The two Dotdeb related lines

```
deb http://packages.dotdeb.org stable all  
deb-src http://packages.dotdeb.org stable all
```

The last thing to consider is whether your list of packages is up to date. If you have not updated it recently, you need to run apt-get update before anything else. This way, you will be using the most recent stable version of the Apache and PHP packages.

Now that everything is in place, you can use the following example to install Apache and PHP:

Example. Debian Install Example with Apache 1.3

```
# apt-get install libapache-mod-php4
```

APT will automatically install the PHP 4 module for Apache 1.3, and all its dependencies and then activate it. If you're not asked to restart Apache during the install process, you'll have to do it manually :

Example. Stopping and starting Apache once PHP 4 is installed

```
# /etc/init.d/apache stop  
# /etc/init.d/apache start
```

## Better control on configuration

In the last section, PHP was installed with only core modules. This may not be what you want and you will soon discover that you need more activated modules, like MySQL, cURL, GD, etc.

When you compile PHP from source yourself, you need to activate modules via the configure command. With APT, you just have to install additional packages. They're all named 'php4-\*' (or 'php5-\*' if you installed PHP 5 from a third party repository).

Example. Getting the list of PHP additional packages

```
# dpkg -l 'php4-*'
```

As you can see from the last output, there's a lot of PHP modules that you can install (excluding the php4-cgi, php4-cli or php4-peach special packages). Look at them closely and choose what you need. If you choose a module and you do not have the proper libraries, APT will automatically install all the dependencies for you.

If you choose to add the MySQL, cURL and GD support to PHP the command will look something like this:

Example. Install PHP with MySQL, cURL and GD

```
# apt-get install php4-mysql php4-curl php4-gd
```

APT will automatically add the appropriate lines to your different **php.ini** (/etc/php4/apache/php.ini, /etc/php4/cgi/php.ini, etc).

Example. These lines activate MySQL, cURL and GD into PHP

```
extension=mysql.so  
extension=curl.so  
extension=gd.so
```

You'll only have to stop/start Apache as previously to activate the modules.

## Common Problems

- If you see the PHP source instead of the result the script should produce, APT has probably not included /etc/apache/conf.d/php4 in your Apache 1.3 configuration. Please

ensure that the following line is present in your `/etc/apache/httpd.conf` file then stop/start Apache:

Example . This line activates PHP 4 into Apache

```
# Include /etc/apache/conf.d/
```

- If you installed an additional module and if its functions are not available in your scripts, please ensure that the appropriate line is present in your `php.ini`, as seen before. APT may fail during the installation of the additional module, due to a confusing debconf configuration.

## Installation on Windows systems

This section applies to Windows 98/Me and Windows NT/2000/XP/2003. PHP will not work on 16 bit platforms such as Windows 3.1 and sometimes we refer to the supported Windows platforms as Win32. Windows 95 is no longer supported as of PHP 4.3.0.

There are two main ways to install PHP for Windows: either manually or by using the installer.

If you have Microsoft Visual Studio, you can also build PHP from the original source code.

Once you have PHP installed on your Windows system, you may also want to load various extensions for added functionality.

Warning: There are several all-in-one installers over the Internet, but none of those are endorsed by PHP.net, as we believe that the manual installation is the best choice to have your system secure and optimized.

### Windows Installer

The Windows PHP installer is available from the downloads page at <http://www.php.net/downloads.php>. This installs the CGI version of PHP and for IIS, PWS, and Xitami, it configures the web server as well. The installer does not include any extra external PHP extensions (`php_*.dll`) as you'll only find those in the Windows Zip Package and PECL downloads.

Note: While the Windows installer is an easy way to make PHP work, it is restricted in many aspects as, for example, the automatic setup of extensions is not supported. Use of the installer isn't the preferred method for installing PHP.

First, install your selected HTTP (web) server on your system, and make sure that it works.

Run the executable installer and follow the instructions provided by the installation wizard. Two types of installation are supported - standard, which provides sensible defaults for all the settings it can, and advanced, which asks questions as it goes along.

The installation wizard gathers enough information to set up the `php.ini` file, and configure certain web servers to use PHP. One of the web servers the PHP installer does not configure for is Apache, so you'll need to configure it manually.

Once the installation has completed, the installer will inform you if you need to restart your system, restart the server, or just start using PHP.

Warning: Be aware, that this setup of PHP is not secure. If you would like to have a secure PHP setup, you'd better go on the manual way, and set every option carefully. This automatically working setup gives you an instantly working PHP installation, but it is not meant to be used on online servers.

## Manual Installation Steps

This install guide will help you manually install and configure PHP with a web server on Microsoft Windows. To get started you'll need to download the zip binary distribution from the downloads page at <http://www.php.net/downloads.php>.

Although there are many all-in-one installation kits, and we also distribute a PHP installer for Microsoft Windows, we recommend you take the time to setup PHP yourself as this will provide you with a better understanding of the system, and enables you to install PHP extensions easily when needed.

**Upgrading from a previous PHP version:** Previous editions of the manual suggest moving various ini and DLL files into your SYSTEM (i.e. C:\WINDOWS) folder and while this simplifies the installation procedure it makes upgrading difficult. We advise you remove all of these files (like php.ini and PHP related DLLs from the Windows SYSTEM folder) before moving on with a new PHP installation. Be sure to backup these files as you might break the entire system. The old php.ini might be useful in setting up the new PHP as well. And as you'll soon learn, the preferred method for installing PHP is to keep all PHP related files in one directory and have this directory available to your systems PATH.

**MDAC requirements:** If you use Microsoft Windows 98/NT4 download the latest version of the Microsoft Data Access Components (MDAC) for your platform. MDAC is available at <http://msdn.microsoft.com/data/>. This requirement exists because ODBC is built into the distributed Windows binaries.

The following steps should be completed on all installations before any server specific instructions are performed:

Extract the distribution file into a directory of your choice. If you are installing PHP 4, extract to C:\, as the zip file expands to a foldername like php-4.3.7-Win32. If you are installing PHP 5, extract to C:\php as the zip file doesn't expand as in PHP 4. You may choose a different location but do not have spaces in the path (like C:\Program Files\PHP) as some web servers will crash if you do.

The directory structure extracted from the zip is different for PHP versions 4 and 5 and look like as follows:

PHP 4 package structure

```
c:\php
|
+--cli
| |
| |-php.exe      -- CLI executable - ONLY for commandline scripting
|
+--dlls          -- support DLLs required by some extensions
| |
| |-expat.dll
| |
| |-fdftk.dll
| |
| |...
|
+--extensions    -- extension DLLs for PHP
| |
| |-php_bz2.dll
| |
| |-php_cpdf.dll
| |
```

```

| |-
| |
+--mibs          -- support files for SNMP
|
+--openssl       -- support files for Openssl
|
+--pdf-related   -- support files for PDF
|
+--sapi          -- SAPI (server module support) DLLs
| |
| |-php4apache.dll
| |
| |-php4apache2.dll
| |
| |-
| |
+--PEAR          -- initial copy of PEAR
|
|
|-go-pear.bat    -- PEAR setup script
|
|-
| |-
|-php.exe        -- CGI executable
|
|-
| |-
|-php.ini-dist   -- default php.ini settings
|
|-php.ini-recommended -- recommended php.ini settings
|
|-php4ts.dll     -- core PHP DLL
|
|-
| |-

```

Or:

PHP 5 package structure

```

c:\php
|
+--dev
| |
| |-php5ts.lib
|
+--ext          -- extension DLLs for PHP
| |
| |-php_bz2.dll
| |
| |-php_cpdf.dll
| |
| |-
| |
+--extras
| |
| +--mibs          -- support files for SNMP
| |
| +--openssl       -- support files for Openssl
| |
| +--pdf-related   -- support files for PDF
| |

```

```

| |-mime.magic
|
| +-pear          -- initial copy of PEAR
|
|
| -go-pear.bat   -- PEAR setup script
|
| -fdftk.dll
|
| ...
|
| -php-cgi.exe   -- CGI executable
|
| -php-win.exe    -- executes scripts without an opened command prompt
|
| -php.exe        -- CLI executable - ONLY for command line scripting
|
| ...
|
| -php.ini-dist   -- default php.ini settings
|
| -php.ini-recommended -- recommended php.ini settings
|
| -php5activescript.dll
|
| -php5apache.dll
|
| -php5apache2.dll
|
| ...
|
| -php5ts.dll     -- core PHP DLL
|
| ...

```

Notice the differences and similarities. Both PHP 4 and PHP 5 have a CGI executable, a CLI executable, and server modules, but they are located in different folders and/or have different names. While PHP 4 packages have the server modules in the sapi folder, PHP 5 distributions have no such directory and instead they're in the PHP folder root. The supporting DLLs for the PHP 5 extensions are also not in a separate directory.

Note: In PHP 4, you should move all files located in the dll and sapi folders to the main folder (e.g. C:\php).

Here is a list of server modules shipped with PHP 4 and PHP 5:

- sapi/php4activescript.dll (php5activescript.dll) - ActiveScript engine, allowing you to embed PHP in your Windows applications.
- sapi/php4apache.dll (php5apache.dll) - Apache 1.3.x module.
- sapi/php4apache2.dll (php5apache2.dll) - Apache 2.0.x module.
- sapi/php4isapi.dll (php5isapi.dll) - ISAPI Module for ISAPI compliant web servers like IIS 4.0/PWS 4.0 or newer.
- sapi/php4nsapi.dll (php5nsapi.dll) - Sun/iPlanet/Netscape server module.
- sapi/php4pi3web.dll (no equivalent in PHP 5) - Pi3Web server module.

Server modules provide significantly better performance and additional functionality compared to the CGI binary. The CLI version is designed to let you use PHP for command line scripting. More information about CLI is available in the chapter about using PHP from the command line.

Warning: The SAPI modules have been significantly improved as of the 4.1 release, however, in older systems you may encounter server errors or other server modules failing, such as ASP.

The CGI and CLI binaries, and the web server modules all require the php4ts.dll (php5ts.dll) file to be available to them. You have to make sure that this file can be found by your PHP installation. The search order for this DLL is as follows:

- The same directory from where php.exe is called, or in case you use a SAPI module, the web server's directory (e.g. C:\Program Files\Apache Group\Apache2\bin).
- Any directory in your Windows PATH environment variable.

To make php4ts.dll / php5ts.dll available you have three options: copy the file to the Windows system directory, copy the file to the web server's directory, or add your PHP directory, C:\php to the PATH. For better maintenance, we advise you to follow the last option, add C:\php to the PATH, because it will be simpler to upgrade PHP in the future. Read more about how to add your PHP directory to PATH in the corresponding FAQ entry (and then don't forget to restart the computer - logoff isn't enough).

The next step is to set up a valid configuration file for PHP, php.ini. There are two ini files distributed in the zip file, php.ini-dist and php.ini-recommended. We advise you to use php.ini-recommended, because we optimized the default settings in this file for performance, and security. Read this well documented file carefully because it has changes from php.ini-dist that will drastically affect your setup. Some examples are display\_errors being off and magic\_quotes\_gpc being off. In addition to reading these, study the ini settings and set every element manually yourself. If you would like to achieve the best security, then this is the way for you, although PHP works fine with these default ini files. Copy your chosen ini-file to a directory that PHP is able to find and rename it to php.ini. PHP searches for php.ini in the locations described in the Section called The configuration file in Chapter 9 section.

If you are running Apache 2, the simpler option is to use the PHPIniDir directive (read the installation on Apache 2 page), otherwise your best option is to set the PHPRC environment variable. This process is explained in the following FAQ entry.

Note: If you're using NTFS on Windows NT, 2000, XP or 2003, make sure that the user running the web server has read permissions to your php.ini (e.g. make it readable by Everyone).

The following steps are optional:

- Edit your new php.ini file. If you plan to use OmniHTTPd, do not follow the next step. Set the doc\_root to point to your web servers document\_root. For example:

```
doc_root = c:\inetpub\wwwroot // for IIS/PWS  
doc_root = c:\apache\htdocs // for Apache
```

- Choose the extensions you would like to load when PHP starts. See the section about Windows extensions, about how to set up one, and what is already built in. Note that on a new installation it is advisable to first get PHP working and tested without any extensions before enabling them in php.ini.
- On PWS and IIS, you can set the browscap configuration setting to point to:  
c:\windows\system\inetsrv\browscap.ini on Windows 9x/Me,  
c:\winnt\system32\inetsrv\browscap.ini on NT/2000, and  
c:\windows\system32\inetsrv\browscap.ini on XP. For an up-to-date browscap.ini, read the following FAQ.

PHP is now setup on your system. The next step is to choose a web server, and enable it to run PHP.

## ActiveScript

This section contains notes specific to the ActiveScript installation. ActiveScript is a windows only SAPI that enables you to use PHP script in any ActiveScript compliant host, like Windows Script Host, ASP/ASP.NET, Windows Script Components or Microsoft Scriptlet control.

As of PHP 5.0.1, ActiveScript has been moved to the PECL repository. You may download this PECL extension DLL from the PHP Downloads page or at <http://snaps.php.net/>.

Note: You should read the manual installation steps first!

After installing PHP, you should download the ActiveScript DLL (php5activescript.dll) and place it in the main PHP folder (e.g. C:\php).

After having all the files needed, you must register the DLL on your system. To achieve this, open a Command Prompt window (located in the Start Menu). Then go to your PHP directory by typing something like cd C:\php. To register the DLL just type regsvr32 php5activescript.dll.

To test if ActiveScript is working, create a new file, named test.wsf (the extension is very important) and type:

```
<job id="test">  
  
<script language="PHPScript">  
 $WScript->Echo("Hello World!");  
</script>  
  
</job>
```

Save and double-click on the file. If you receive a little window saying "Hello World!" you're done.

Note: In PHP 4, the engine was named 'ActivePHP', so if you are using PHP 4, you should replace 'PHPScript' with 'ActivePHP' in the above example.

Note: ActiveScript doesn't use the default php.ini file. Instead, it will look only in the same directory as the .exe that caused it to load. You should create php-activescript.ini and place it in that folder, if you wish to load extensions, etc.

## Microsoft IIS / PWS

This section contains notes and hints specific to IIS (Microsoft Internet Information Server).

### General considerations for all installations of PHP with IIS or PWS

First, read the Manual Installation Instructions. Do not skip this step as it provides crucial information for installing PHP on Windows.

CGI users must set the cgi.force\_redirect PHP directive to 0 inside php.ini. Read the faq on cgi.force\_redirect for important details. Also, CGI users may want to set the cgi.redirect\_status\_env directive. When using directives, be sure these directives aren't commented out inside php.ini.

The PHP 4 CGI is named php.exe while in PHP 5 it's php-cgi.exe. In PHP 5, php.exe is the CLI, and not the CGI.

Modify the Windows PATH environment variable to include the PHP directory. This way the PHP DLL files, PHP executables, and php.ini can all remain in the PHP directory without cluttering up the Windows system directory. For more details, see the FAQ on Setting the PATH.

The IIS user (usually IUSR\_MACHINENAME) needs permission to read various files and directories, such as php.ini, docroot, and the session tmp directory.

Be sure the extension\_dir and doc\_root PHP directives are appropriately set in php.ini. These directives depend on the system that PHP is being installed on. In PHP 4, the extension\_dir is extensions while with PHP 5 it's ext. So, an example PHP 5 extensions\_dir value is "c:\php\ext" and an example IIS doc\_root value is "c:\Inetpub\wwwroot".

PHP extension DLL files, such as php\_mysql.dll and php\_curl.dll, are found in the zip package of the PHP download (not the PHP installer). In PHP 5, many extensions are part of PECL and can be downloaded in the "Collection of PECL modules" package. Files such as php\_zip.dll and php\_ssh2.dll. Download PHP files here.

When defining the executable, the 'check that file exists' box may also be checked. For a small performance penalty, the IIS (or PWS) will check that the script file exists and sort out authentication before firing up PHP. This means that the web server will provide sensible 404 style error messages instead of CGI errors complaining that PHP did not output any data.

## Windows NT/200x/XP and IIS 4 or newer

PHP may be installed as a CGI binary, or with the ISAPI module. In either case, you need to start the Microsoft Management Console (may appear as 'Internet Services Manager', either in your Windows NT 4.0 Option Pack branch or the Control Panel=>Administrative Tools under Windows 2000/XP). Then right click on your Web server node (this will most probably appear as 'Default Web Server'), and select 'Properties'.

If you want to use the CGI binary, do the following:

Under 'Home Directory', 'Virtual Directory', or 'Directory', do the following:

Change the Execute Permissions to 'Scripts only'

Click on the 'Configuration' button, and choose the Application Mappings tab. Click Add and set the Executable path to the appropriate CGI file. An example PHP 5 value is: C:\php\php-cgi.exe Supply .php as the extension. Leave 'Method exclusions' blank, and check the 'Script engine' checkbox. Now, click OK a few times.

Set up the appropriate security. (This is done in Internet Service Manager), and if your NT Server uses NTFS file system, add execute rights for IUSR\_ to the directory that contains php.exe / php-cgi.exe.

To use the ISAPI module, do the following:

- If you don't want to perform HTTP Authentication using PHP, you can (and should) skip this step. Under ISAPI Filters, add a new ISAPI filter. Use PHP as the filter name, and supply a path to the php4isapi.dll / php5isapi.dll.
- Under 'Home Directory', 'Virtual Directory', or 'Directory', do the following:
  - Change the Execute Permissions to 'Scripts only'
  - Click on the 'Configuration' button, and choose the Application Mappings tab. Click Add and set the Executable path to the appropriate ISAPI DLL. An example PHP 5 value is: C:\php\php5isapi.dll Supply .php as the extension. Leave 'Method exclusions' blank, and check the 'Script engine' checkbox. Now, click OK a few times.
  - Stop IIS completely (NET STOP iisadmin)
  - Start IIS again (NET START w3svc)

With IIS 6 (2003 Server), open up the IIS Manager, go to Web Service Extensions, choose "Add a new Web service extension", enter in a name such as PHP, choose the Add button and for the value browse to either the ISAPI file (php4isapi.dll or php5isapi.dll) or CGI (php.exe or php-cgi.exe) then check "Set extension status to Allowed" and click OK.

In order to use index.php as a default content page, do the following: From within the Documents tab, choose Add. Type in index.php and click OK. Adjust the order by choosing Move Up or Move Down. This is similar to setting DirectoryIndex with Apache.

The steps above must be repeated for each extension that is to be associated with PHP scripts. .php is the most common although .php3 may be required for legacy applications.

If you experience 100% CPU usage after some time, turn off the IIS setting Cache ISAPI Application.

## Windows and PWS 4

PWS 4 does not support ISAPI, only PHP CGI should be used.

- Edit the enclosed pws-php4cgi.reg / pws-php5cgi.reg file (look into the SAPI folder for PHP 4, or in the main folder for PHP 5) to reflect the location of your php.exe / php-cgi.exe. Backslashes should be escaped, for example:  
*[HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="C:\php\php.exe"* (change to C:\php\php-cgi.exe if you are using PHP 5)  
Now merge this registry file into your system; you may do this by double-clicking it.
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

## Windows and PWS/IIS 3

The recommended method for configuring these servers is to use the REG file included with the distribution (pws-php4cgi.reg in the SAPI folder for PHP 4, or pws-php5cgi.reg in the main folder for PHP 5). You may want to edit this file and make sure the extensions and PHP install directories match your configuration. Or you can follow the steps below to do it manually.

Warning: These steps involve working directly with the Windows registry. One error here can leave your system in an unstable state. We highly recommend that you back up your registry first. The PHP Development team will not be held responsible if you damage your registry.

- Run Regedit.
- Navigate to: HKEY\_LOCAL\_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap.
- On the edit menu select: New->String Value.
- Type in the extension you wish to use for your php scripts. For example .php
- Double click on the new string value and enter the path to php.exe in the value data field.  
ex: C:\php\php.exe "%s" %s for PHP 4, or C:\php\php-cgi.exe "%s" %s for PHP 5.
- Repeat these steps for each extension you wish to associate with PHP scripts.

The following steps do not affect the web server installation and only apply if you want your PHP scripts to be executed when they are run from the command line (ex. run C:\myscripts\test.php) or by double clicking on them in a directory viewer window. You may wish to skip these steps as you might prefer the PHP files to load into a text editor when you double click on them.

- Navigate to: HKEY\_CLASSES\_ROOT
- On the edit menu select: New->Key.

- Name the key to the extension you setup in the previous section. ex: .php
- Highlight the new key and in the right side pane, double click the "default value" and enter phpfile.
- Repeat the last step for each extension you set up in the previous section.
- Now create another New->Key under HKEY\_CLASSES\_ROOT and name it phpfile.
- Highlight the new key phpfile and in the right side pane, double click the "default value" and enter PHP Script.
- Right click on the phpfile key and select New->Key, name it Shell.
- Right click on the Shell key and select New->Key, name it open.
- Right click on the open key and select New->Key, name it command.
- Highlight the new key command and in the right side pane, double click the "default value" and enter the path to php.exe. ex: c:\php\php.exe -q %1. (don't forget the %1).
- Exit Regedit.
- If using PWS on Windows, reboot to reload the registry.

PWS and IIS 3 users now have a fully operational system. IIS 3 users can use a nifty tool from Steven Genusa to configure their script maps.

## Apache 1.3.x on Microsoft Windows

This section contains notes and hints specific to Apache 1.3.x installs of PHP on Microsoft Windows systems. There are also instructions and notes for Apache 2 on a separate page.

Note: Please read the manual installation steps first!

There are two ways to set up PHP to work with Apache 1.3.x on Windows. One is to use the CGI binary (php.exe for PHP 4 and php-cgi.exe for PHP 5), the other is to use the Apache Module DLL. In either case you need to edit your httpd.conf to configure Apache to work with PHP, and then restart the server.

It is worth noting here that now the SAPI module has been made more stable under Windows, we recommend it's use above the CGI binary, since it is more transparent and secure.

Although there can be a few variations of configuring PHP under Apache, these are simple enough to be used by the newcomer. Please consult the Apache Documentation for further configuration directives.

After changing the configuration file, remember to restart the server, for example, NET STOP APACHE followed by NET START APACHE, if you run Apache as a Windows Service, or use your regular shortcuts.

Note: Remember that when adding path values in the Apache configuration files on Windows, all backslashes such as c:\directory\file.ext must be converted to forward slashes, as c:/directory/file.ext. A trailing slash may also be necessary for directories.

### Installing as an Apache module

You should add the following lines to your Apache httpd.conf file:

#### **Example 6-3. PHP as an Apache 1.3.x module**

This assumes PHP is installed to C:\php. Adjust the path if this is not the case.

For PHP 4:

```
# Add to the end of the LoadModule section
```

```
# Don't forget to copy this file from the sapi directory!
LoadModule php4_module "C:/php/php4apache.dll"

# Add to the end of the AddModule section
AddModule mod_php4.c
```

For PHP 5:

```
# Add to the end of the LoadModule section
LoadModule php5_module "C:/php/php5apache.dll"

# Add to the end of the AddModule section
AddModule mod_php5.c
```

For both:

```
# Add this line inside the <IfModule mod_mime.c> conditional brace
AddType application/x-httpd-php .php

# For syntax highlighted .phps files, also add
AddType application/x-httpd-php-source .phps
```

Installing as a CGI binary

If you unzipped the PHP package to C:\php\ as described in the Manual Installation Steps section, you need to insert these lines to your Apache configuration file to set up the CGI binary:

Example 6-4. PHP and Apache 1.3.x as CGI

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php

# For PHP 4
Action application/x-httpd-php "/php/php.exe"

# For PHP 5
Action application/x-httpd-php "/php/php-cgi.exe"

# specify the directory where php.ini is
SetEnv PHPRC C:/php
```

Note that the second line in the list above can be found in the actual versions of httpd.conf, but it is commented out. Remember also to substitute the c:/php/ for your actual path to PHP.

Warning: By using the CGI setup, your server is open to several possible attacks. Please read our CGI security section to learn how to defend yourself from those attacks.

If you would like to present PHP source files syntax highlighted, there is no such convenient option as with the module version of PHP. If you chose to configure Apache to use PHP as a CGI binary, you will need to use the highlight\_file() function. To do this simply create a PHP script file and add this code: <?php highlight\_file('some\_php\_script.php'); ?>

## Apache 2.0.x on Microsoft Windows

This section contains notes and hints specific to Apache 2.0.x installs of PHP on Microsoft Windows systems. We also have instructions and notes for Apache 1.3.x users on a separate page.

Note: You should read the manual installation steps first!

Apache 2.2.x Support: Users of Apache 2.2.x may use the documentation below except the appropriate DLL file is named `php5apache2_2.dll` and it only exists as of PHP 5.2.0. See also <http://snaps.php.net/>

Warning: We do not recommend using a threaded MPM in production with Apache2. Use the prefork MPM instead, or use Apache1. For information on why, read the related FAQ entry on using Apache2 with a threaded MPM

. You are highly encouraged to take a look at the Apache Documentation to get a basic understanding of the Apache 2.0.x Server. Also consider to read the Windows specific notes for Apache 2.0.x before reading on here.

PHP and Apache 2.0.x compatibility notes: The following versions of PHP are known to work with the most recent version of Apache 2.0.x:

- PHP 4.3.0 or later available at <http://www.php.net/downloads.php>.
- the latest stable development version. Get the source code <http://snaps.php.net/php5-latest.tar.gz> or download binaries for Windows <http://snaps.php.net/win32/php5-win32-latest.zip>.
- a prerelease version downloadable from <http://qa.php.net/>.
- you have always the option to obtain PHP through [anonymous CVS](#).

These versions of PHP are compatible to Apache 2.0.40 and later.

Apache 2.0 SAPI-support started with PHP 4.2.0. PHP 4.2.3 works with Apache 2.0.39, don't use any other version of Apache with PHP 4.2.3. However, the recommended setup is to use PHP 4.3.0 or later with the most recent version of Apache2.

All mentioned versions of PHP will work still with Apache 1.3.x.

Warning: Apache 2.0.x is designed to run on Windows NT 4.0, Windows 2000 or Windows XP. At this time, support for Windows 9x is incomplete. Apache 2.0.x is not expected to work on those platforms at this time.

Download the most recent version of Apache 2.0.x and a fitting PHP version. Follow the Manual Installation Steps and come back to go on with the integration of PHP and Apache.

There are two ways to set up PHP to work with Apache 2.0.x on Windows. One is to use the CGI binary the other is to use the Apache module DLL. In either case you need to edit your `httpd.conf` to configure Apache to work with PHP and then restart the server.

Note: Remember that when adding path values in the Apache configuration files on Windows, all backslashes such as `c:\directory\file.ext` must be converted to forward slashes, as `c:/directory/file.ext`. A trailing slash may also be necessary for directories.

## Installing as a CGI binary

You need to insert these three lines to your Apache `httpd.conf` configuration file to set up the CGI binary:

Example . PHP and Apache 2.0 as CGI

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httdp-php .php

# For PHP 4
Action application/x-httdp-php "/php/php.exe"

# For PHP 5
Action application/x-httdp-php "/php/php-cgi.exe"
```

Warning: By using the CGI setup, your server is open to several possible attacks. Please read our CGI security section to learn how to defend yourself from those attacks.

#### Installing as an Apache module

You need to insert these two lines to your Apache httpd.conf configuration file to set up the PHP module for Apache 2.0:

#### Example 6-6. PHP and Apache 2.0 as Module

```
# For PHP 4 do something like this:
LoadModule php4_module "c:/php/php4apache2.dll"
# Don't forget to copy the php4apache2.dll file from the sapi directory!
AddType application/x-httdp-php .php

# For PHP 5 do something like this:
LoadModule php5_module "c:/php/php5apache2.dll"
AddType application/x-httdp-php .php

# configure the path to php.ini
PHPIniDir "C:/php"
```

Note: Remember to substitute your actual path to PHP for the c:/php/ in the above examples. Take care to use either php4apache2.dll or php5apache2.dll in your LoadModule directive and not php4apache.dll or php5apache.dll as the latter ones are designed to run with Apache 1.3.x.

Warning: Don't mix up your installation with DLL files from different PHP versions. You have the only choice to use the DLL's and extensions that ship with your downloaded PHP version.

# **ASP.NET**

**ASP.NET is the latest version of Microsoft's Active Server Pages technology (ASP).**

To understand asp.net you should have a basic understanding of the following:

- WWW, HTML and the basics of building Web pages
- Scripting languages like JavaScript or VBScript
- The basics of server side scripting

## **What is ASP?**

ASP is a server side scripting technology that enables scripts (embedded in web pages) to be executed by an Internet server.

- ASP is a Microsoft Technology
- ASP stands for Active Server Pages
- ASP is a program that runs inside IIS
- IIS stands for Internet Information Services
- IIS comes as a free component with Windows 2000
- IIS is also a part of the Windows NT 4.0 Option Pack
- The Option Pack can be downloaded from Microsoft
- PWS is a smaller - but fully functional - version of IIS
- PWS can be found on your Windows 95/98 CD

## **What is an ASP File?**

- An ASP file is just the same as an HTML file
- An ASP file can contain text, HTML, XML, and scripts
- Scripts in an ASP file are executed on the server
- An ASP file has the file extension ".asp"

## **How Does it Work?**

- When a browser requests an HTML file, the server returns the file
- When a browser requests an ASP file, IIS passes the request to the ASP engine on the server
- The ASP engine reads the file, line by line, and executes the scripts in the file
- Finally, the ASP file is returned to the browser as plain HTML

Before you study ASP .NET, it would help to have a basic understanding of Microsoft's ASP technology.

## **What is ASP+?**

ASP+ is the same as ASP.NET.

ASP+ is just an early name used by Microsoft when they developed ASP.NET.

## **What is ASP.NET?**

ASP 3.0 is the latest version of ASP, but there will never be an ASP 4.0 version.

ASP.NET is the next generation ASP, but it's not an upgraded version of ASP. ASP.NET is an entirely new paradigm for server-side ASP scripting.

ASP.NET is a part of the .NET Framework. Microsoft spent three years rewriting ASP.NET from the ground up, and ASP.NET is not fully backward compatible with ASP 3.0.

You can read more about the differences between ASP and ASP.NET in the next chapter of this tutorial.

## .NET Framework

The .NET Framework is the infrastructure for the Microsoft .NET platform.

The .NET Framework is an environment for building, deploying, and running Web applications and Web Services.

The .NET Framework contains a common language runtime and common class libraries - like ADO.NET, ASP.NET and Windows Forms - to provide advanced standard services that can be integrated into a variety of computer systems.

The .NET Framework provides a feature-rich application environment, simplified development and easy integration between a number of different development languages.

The .NET Framework is language neutral. Currently it supports C++, C#, Visual Basic, and JScript (Microsoft's version of JavaScript).

Microsoft's Visual Studio.NET is a common development environment for the .NET Framework.

## Differences between ASP and ASP .NET

ASP .NET has better language support, a large set of new controls and XML based components, and better user authentication. ASP .NET provides increased performance by running compiled code. ASP .NET code is not fully backward compatible with ASP.

## New in ASP .NET

- Better language support
- Programmable controls
- Event-driven programming
- XML-based components
- User authentication, with accounts and roles
- Higher scalability
- Increased performance - Compiled code
- Easier configuration and deployment
- Not fully ASP compatible

## Language Support

ASP .NET uses the new ADO .NET.

ASP .NET supports full Visual Basic, not VBScript.

ASP .NET supports C# (C sharp) and C++.

ASP .NET supports JScript as before.

## ASP .NET Controls

ASP .NET contains a large set of HTML controls. Almost all HTML elements on a page can be defined as ASP .NET control objects that can be controlled by scripts.

ASP .NET also contains a new set of object oriented input controls, like programmable list boxes and validation controls.

A new data grid control supports sorting, data paging, and everything you expect from a dataset control.

## **Event Aware Controls**

All ASP .NET objects on a Web page can expose events that can be processed by ASP .NET code.

Load, Click and Change events handled by code makes coding much simpler and much better organized.

## **ASP .NET Components**

ASP .NET components are heavily based on XML. Like the new AD Rotator, that uses XML to store advertisement information and configuration.

## **User Authentication**

ASP .NET supports forms-based user authentication, including cookie management and automatic redirecting of unauthorized logins.

(You can still do your custom login page and custom user checking).

## **User Accounts and Roles**

ASP .NET allows for user accounts and roles, to give each user (with a given role) access to different server code and executables.

## **High Scalability**

Much has been done with ASP .NET to provide greater scalability.

Server to server communication has been greatly enhanced, making it possible to scale an application over several servers. One example of this is the ability to run XML parsers, XSL transformations and even resource hungry session objects on other servers.

## **Compiled Code**

The first request for an ASP .NET page on the server will compile the ASP .NET code and keep a cached copy in memory. The result of this is greatly increased performance.

## **Easy Configuration**

Configuration of ASP .NET is done with plain text files.

Configuration files can be uploaded or changed while the application is running. No need to restart the server. No more metabase or registry puzzle.

## **Easy Deployment**

No more server restart to deploy or replace compiled code. ASP .NET simply redirects all new requests to the new code.

## **Compatibility**

ASP .NET is not fully compatible with earlier versions of ASP, so most of the old ASP code will need some changes to run under ASP .NET.

To overcome this problem, ASP .NET uses a new file extension ".aspx". This will make ASP .NET applications able to run side by side with standard ASP applications on the same server.

# Installing ASP.NET

**ASP.NET is easy to install. Just follow the instructions below.**

## What You Need

### A Windows Computer

ASP.NET is a Microsoft technology. To run ASP.NET you need a computer capable of running Windows.

### Windows 2000 or XP

If you are serious about developing ASP.NET applications you should install Windows 2000 Professional or Windows XP Professional.

In both cases, make sure you install the Internet Information Services (IIS) from the Add/Remove Windows components dialog.

### Service Packs and Updates

Before ASP.NET can be installed on your computer, it is necessary to have all relevant service packs and security updates installed.

The easiest way to do this is to activate your Windows Internet Update. When you access the Windows Update page, you will be instructed to install the latest service packs and all critical security updates. For Windows 2000, make sure you install Service Pack 2. I will also recommend that you install Internet Explorer 6.

Read the note about connection speed and download time at the bottom of this page.

### Remove Your Beta Version

If you have a Beta version of ASP.NET installed, we recommend that you completely uninstall it. Or even better: start with a fresh Windows 2000 or XP installation.

## Install .NET

From your Windows Update you can now select to install the Microsoft .NET Framework.

After download, the .NET framework will install itself on your computer - there are no options to select for installation.

**You should now be ready to develop your first ASP.NET application!**

## The .NET Software Development Kit

If you have the necessary bandwidth to download over 130 MB, you might consider downloading the full Microsoft .NET Software Development Kit (SDK).

We fully recommend getting the SDK for learning more about .NET and for the documentation, samples, and tools included.

## Connection Speed and Download Time

If you have a slow Internet connection, you might have problems downloading large files like the service packs, the SDK and the latest version of Internet Explorer.

If download speed is a problem, our best suggestion is to get the latest files from someone else, from a colleague, from a friend, or from one of the CDs that comes with many popular computer magazines. Look for Windows 2000 Service Pack 2, Internet Explorer 6, and the Microsoft .NET Framework.

## ASP.NET - Web Pages

A simple ASP.NET page looks just like an ordinary HTML page.

### Hello W3Schools

To start learning ASP.NET, we will construct a very simple HTML page that will display "Hello W3Schools" in an Internet browser like this:



Hello W3Schools!

### Hello W3Schools in HTML

This code displays the example as an HTML page:

```
<html>
<body bgcolor="yellow">
<center>
<h2>Hello W3Schools!</h2>
</center>
</body>
</html>
```

If you want to try it yourself, save the code in a file called "**firstpage.htm**", and create a link to the file like this: firstpage.htm

### Hello W3Schools in ASP.NET

The simplest way to convert an HTML page into an ASP.NET page is to copy the HTML file to a new file with an **.aspx** extension.

This code displays our example as an ASP.NET page:

```
<html>
<body bgcolor="yellow">
<center>
<h2>Hello W3Schools!</h2>
</center>
</body>
</html>
```

If you want to try it yourself, save the code in a file called "**firstpage.aspx**", and create a link to the file like this: firstpage.aspx

### How Does it Work?

Fundamentally an ASP.NET page is just the same as an HTML page.

An HTML page has the extension .htm. If a browser requests an HTML page from the server, the server sends the page to the browser without any modifications.

An ASP.NET page has the extension .aspx. If a browser requests an ASP.NET page, the server processes any executable code in the page, before the result is sent back to the browser.

The ASP.NET page above does not contain any executable code, so nothing is executed. In the next examples we will add some executable code to the page to demonstrate the difference between static HTML pages and dynamic ASP pages.

## Classic ASP

Active Server Pages (ASP) has been around for several years. With ASP, executable code can be placed inside HTML pages.

Previous versions of ASP (before ASP .NET) are often called Classic ASP.

ASP .NET is not fully compatible with Classic ASP, but most Classic ASP pages will work fine as ASP .NET pages, with only minor changes.

If you want to learn more about Classic ASP, please visit our [ASP Tutorial](#).

## Dynamic Page in Classic ASP

To demonstrate how ASP can display pages with dynamic content, we have added some executable code (in red) to the previous example:

```
<html>
<body bgcolor="yellow">
<center>
<h2>Hello W3Schools!</h2>
<p><%Response.Write(now())%></p>
</center>
</body>
</html>
```

The code inside the <% --%> tags is executed on the server.

Response.Write is ASP code for writing something to the HTML output stream.

Now() is a function returning the servers current date and time.

If you want to try it yourself, save the code in a file called "**dynpage.asp**", and create a link to the file like this: dynpage.asp

## Dynamic Page in ASP .NET

This following code displays our example as an ASP .NET page:

```
<html>
<body bgcolor="yellow">
<center>
<h2>Hello W3Schools!</h2>
<p><%Response.Write(now())%></p>
</center>
</body>
</html>
```

If you want to try it yourself, save the code in a file called "**dynpage.aspx**", and create a link to the file like this: dynpage.aspx

## ASP .NET vs Classic ASP

The previous examples didn't demonstrate any differences between ASP .NET and Classic ASP.

As you can see from the two latest examples there are no differences between the two ASP and ASP .NET pages.

In the next chapters you will see how server controls make ASP. NET more powerful than Classic ASP.

## ASP.NET - Server Controls

**Server controls are tags that are understood by the server.**

### Limitations in Classic ASP

The listing below was copied from the previous chapter:

```
<html>
<body bgcolor="yellow">
<center>
<h2>Hello W3Schools!</h2>
<p><%Response.Write(now())%></p>
</center>
</body>
</html>
```

The code above illustrates a limitation in Classic ASP: The code block has to be placed where you want the output to appear.

With Classic ASP it is impossible to separate executable code from the HTML itself. This makes the page difficult to read, and difficult to maintain.

## ASP.NET - Server Controls

ASP.NET has solved the "spaghetti-code" problem described above with server controls.

Server controls are tags that are understood by the server.

There are three kinds of server controls:

- HTML Server Controls - Traditional HTML tags
- Web Server Controls - New ASP.NET tags
- Validation Server Controls - For input validation

## ASP.NET - HTML Server Controls

HTML server controls are HTML tags understood by the server.

HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a `runat="server"` attribute to the HTML element. This attribute indicates that the element should be treated as a server control. The `id` attribute is added to identify the server control. The `id` reference can be used to manipulate the server control at run time.

**Note:** All HTML server controls must be within a <form> tag with the runat="server" attribute. The runat="server" attribute indicates that the form should be processed on the server. It also indicates that the enclosed controls can be accessed by server scripts.

In the following example we declare an HtmlAnchor server control in an .aspx file. Then we manipulate the HRef attribute of the HtmlAnchor control in an event handler (an event handler is a subroutine that executes code for a given event). The Page\_Load event is one of many events that ASP.NET understands:

```
<script runat="server">
Sub Page_Load
link1.HRef="http://www.w3schools.com"
End Sub
</script>
<html>
<body>
<form runat="server">
<a id="link1" runat="server">Visit W3Schools!</a>
</form>
</body>
</html>
```

The executable code itself has been moved outside the HTML.

## ASP.NET - Web Server Controls

Web server controls are special ASP.NET tags understood by the server.

Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.

The syntax for creating a Web server control is:

```
<asp:control_name id="some_id" runat="server" />
```

In the following example we declare a Button server control in an .aspx file. Then we create an event handler for the Click event which changes the text on the button:

```
<script runat="server">
Sub submit(Source As Object, e As EventArgs)
button1.Text="You clicked me!"
End Sub
</script>
<html>
<body>
<form runat="server">
<asp:Button id="button1" Text="Click me!" 
runat="server" OnClick="submit"/>
</form>
</body>
</html>
```

## ASP.NET - Validation Server Controls

Validation server controls is used to validate user-input. If the user-input does not pass validation, it will display an error message to the user.

Each validation control performs a specific type of validation (like validating against a specific value or a range of values).

By default, page validation is performed when a Button, ImageButton, or LinkButton control is clicked. You can prevent validation when a button control is clicked by setting the CausesValidation property to false.

The syntax for creating a Validation server control is:

```
<asp:control_name id="some_id" runat="server" />
```

In the following example we declare one TextBox control, one Button control, and one RangeValidator control in an .aspx file. If validation fails, the text "The value must be from 1 to 100!" will be displayed in the RangeValidator control:

```
<html>
<body>
<form runat="server">
Enter a number from 1 to 100:
<asp:TextBox id="tbox1" runat="server" />
<br /><br />
<asp:Button Text="Submit" runat="server" />
<br />
<asp:RangeValidator
ControlToValidate="tbox1"
MinimumValue="1"
MaximumValue="100"
Type="Integer"
EnableClientScript="false"
Text="The value must be from 1 to 100!"
runat="server" />
</form>
</body>
</html>
```

## ASP.NET – Events

**An Event Handler is a subroutine that executes code for a given event.**

### ASP.NET - Event Handlers

Look at the following code:

```
<%
lbl1.Text="The date and time is " & now()
%>
<html>
<body>
<form runat="server">
<h3><asp:label id="lbl1" runat="server" /></h3>
</form>
</body>
</html>
```

When will the code above be executed? The answer is: "You don't know..."

## The Page\_Load Event

The Page\_Load event is one of many events that ASP.NET understands. The Page\_Load event is triggered when a page loads, and ASP.NET will automatically call the subroutine Page\_Load, and execute the code inside it:

```
<script runat="server">
Sub Page_Load
lbl1.Text="The date and time is " & now()
End Sub
</script>
<html>
<body>
<form runat="server">
<h3><asp:label id="lbl1" runat="server" /></h3>
</form>
</body>
</html>
```

**Note:** The Page\_Load event contains no object references or event arguments!

## The Page.IsPostBack Property

The Page\_Load subroutine runs EVERY time the page is loaded. If you want to execute the code in the Page\_Load subroutine only the FIRST time the page is loaded, you can use the Page.IsPostBack property. If the Page.IsPostBack property is false, the page is loaded for the first time, if it is true, the page is posted back to the server (i.e. from a button click on a form):

```
<script runat="server">
Sub Page_Load
if Not Page.IsPostBack then
    lbl1.Text="The date and time is " & now()
end if
End Sub
Sub Submit(s As Object, e As EventArgs)
lbl2.Text="Hello World!"
End Sub
</script>
<html>
<body>
<form runat="server">
<h3><asp:label id="lbl1" runat="server" /></h3>
<h3><asp:label id="lbl2" runat="server" /></h3>
<asp:button text="Submit" onclick="submit" runat="server" />
</form>
</body>
</html>
```

The example above will write the "The date and time is...." message only the first time the page is loaded. When a user clicks on the Submit button, the submit subroutine will write "Hello World!" to the second label, but the date and time in the first label will not change.

## ASP.NET Web Forms

All server controls must appear within a `<form>` tag, and the `<form>` tag must contain the `runat="server"` attribute.

## ASP.NET Web Forms

All server controls must appear within a <form> tag, and the <form> tag must contain the runat="server" attribute. The runat="server" attribute indicates that the form should be processed on the server. It also indicates that the enclosed controls can be accessed by server scripts:

```
<form runat="server">
...HTML + server controls
</form>
```

**Note:** The form is always submitted to the page itself. If you specify an action attribute, it is ignored. If you omit the method attribute, it will be set to method="post" by default. Also, if you do not specify the name and id attributes, they are automatically assigned by ASP.NET.

**Note:** An .aspx page can only contain ONE <form runat="server"> control!

If you select view source in an .aspx page containing a form with no name, method, action, or id attribute specified, you will see that ASP.NET has added these attributes to the form. It looks something like this:

```
<form name="_ctl0" method="post" action="page.aspx" id="_ctl0">
...some code
</form>
```

## Submitting a Form

A form is most often submitted by clicking on a button. The Button server control in ASP.NET has the following format:

```
<asp:Button id="id" text="label" OnClick="sub" runat="server" />
```

The id attribute defines a unique name for the button and the text attribute assigns a label to the button. The onClick event handler specifies a named subroutine to execute.

In the following example we declare a Button control in an .aspx file. A button click runs a subroutine which changes the text on the button:

## ASP .NET Maintaining the ViewState

**You may save a lot of coding by maintaining the ViewState of the objects in your Web Form.**

### Maintaining the ViewState

When a form is submitted in classic ASP, all form values are cleared. Suppose you have submitted a form with a lot of information and the server comes back with an error. You will have to go back to the form and correct the information. You click the back button, and what happens.....ALL form values are CLEARED, and you will have to start all over again! The site did not maintain your ViewState.

When a form is submitted in ASP .NET, the form reappears in the browser window together with all form values. How come? This is because ASP .NET maintains your ViewState. The ViewState indicates the status of the page when submitted to the server. The status is defined through a hidden field placed on each page with a <form runat="server"> control. The source could look something like this:

```
<form name="_ctl0" method="post" action="page.aspx" id="_ctl0">
<input type="hidden" name="__VIEWSTATE"
value="dDwtNTI0ODU5MDE1Ozs+ZBCF2ryjMpeVgUrY2eTj79HNl4Q=" />
....some code
</form>
```

Maintaining the ViewState is the default setting for ASP.NET Web Forms. If you want to NOT maintain the ViewState, include the directive `<%@ Page EnableViewState="false" %>` at the top of an .aspx page or add the attribute `EnableViewState="false"` to any control.

Look at the following .aspx file. It demonstrates the "old" way to do it. When you click on the submit button, the form value will disappear:

```
<html>
<body>
<form action="demo_classicasp.aspx" method="post">
Your name: <input type="text" name="fname" size="20">
<input type="submit" value="Submit">
</form>
<%
dim fname
fname=Request.Form("fname")
If fname<>" " Then
Response.Write("Hello " & fname & "!")
End If
%>
</body>
</html>
```

Here is the new ASP .NET way. When you click on the submit button, the form value will NOT disappear:

```
<script runat="server">
Sub submit(sender As Object, e As EventArgs)
lbl1.Text="Hello " & txt1.Text & "!"
End Sub
</script>
<html>
<body>
<form runat="server">
Your name: <asp:TextBox id="txt1" runat="server" />
<asp:Button OnClick="submit" Text="Submit" runat="server" />
<p><asp:Label id="lbl1" runat="server" /></p>
</form>
</body>
</html>
```

## ASP .NET - The TextBox Control

The TextBox control is used to create a text box where the user can input text.

### The TextBox Control

The TextBox control is used to create a text box where the user can input text.

The example below demonstrates some of the attributes you may use with the TextBox control:

```
<html>
<body>

<form runat="server">

A basic TextBox:
<asp:TextBox id="tbl" runat="server" />
<br /><br />
```

```

A password TextBox:
<asp:TextBox id="tb2" TextMode="password" runat="server" />
<br /><br />

A TextBox with text:
<asp:TextBox id="tb4" Text="Hello World!" runat="server" />
<br /><br />

A multiline TextBox:
<asp:TextBox id="tb3" TextMode="multiline" runat="server" />
<br /><br />

A TextBox with height:
<asp:TextBox id="tb6" rows="5" TextMode="multiline"
runat="server" />
<br /><br />

A TextBox with width:
<asp:TextBox id="tb5" columns="30" runat="server" />

</form>

</body>
</html>

```

## Add a Script

The contents and settings of a TextBox control may be changed by server scripts when a form is submitted. A form can be submitted by clicking on a button or when a user changes the value in the TextBox control.

In the following example we declare one TextBox control, one Button control, and one Label control in an .aspx file. When the submit button is triggered, the submit subroutine is executed. The submit subroutine writes a text to the Label control:

```

<script runat="server">
Sub submit(sender As Object, e As EventArgs)
lbl1.Text="Your name is " & txt1.Text
End Sub
</script>
<html>
<body>
<form runat="server">
Enter your name:
<asp:TextBox id="txt1" runat="server" />
<asp:Button OnClick="submit" Text="Submit" runat="server" />
<p><asp:Label id="lbl1" runat="server" /></p>
</form>
</body>
</html>

```

In the following example we declare one TextBox control and one Label control in an .aspx file. When you change the value in the TextBox and either click outside the TextBox or press the Tab key, the change subroutine is executed. The submit subroutine writes a text to the Label control:

```

<script runat="server">
Sub change(sender As Object, e As EventArgs)
lbl1.Text="You changed text to " & txt1.Text
End Sub
</script>

```

```

<html>
<body>
<form runat="server">
Enter your name:
<asp:TextBox id="txt1" runat="server"
text="Hello World!" 
onTextChanged="change" autopostback="true"/>
<p><asp:Label id="lbl1" runat="server" /></p>
</form>
</body>
</html>

```

## ASP.NET - The Button Control

The Button control is used to display a push button.

### **The Button Control**

The Button control is used to display a push button. The push button may be a submit button or a command button. By default, this control is a submit button.

A submit button does not have a command name and it posts the page back to the server when it is clicked. It is possible to write an event handler to control the actions performed when the submit button is clicked.

A command button has a command name and allows you to create multiple Button controls on a page. It is possible to write an event handler to control the actions performed when the command button is clicked.

The example below demonstrates a simple Button control:

```

<html>
<body>

<form runat="server">
<asp:Button id="b1" Text="Submit" runat="server" />
</form>
</body>
</html>

```

### **Add a Script**

A form is most often submitted by clicking on a button.

In the following example we declare one TextBox control, one Button control, and one Label control in an .aspx file. When the submit button is triggered, the submit subroutine is executed. The submit subroutine writes a text to the Label control:

```

<script runat="server">
Sub submit(sender As Object, e As EventArgs)
lbl1.Text="Your name is " & txt1.Text
End Sub
</script>
<html>
<body>
<form runat="server">
Enter your name:
<asp:TextBox id="txt1" runat="server" />

```

```
<asp:Button OnClick="submit" Text="Submit" runat="server" />
<p><asp:Label id="lbl1" runat="server" /></p>
</form>
</body>
</html>
```

## ASP.NET - Data Binding

We may use data binding to fill lists with selectable items from an imported data source, like a database, an XML file, or a script.

### Data Binding

The following controls are list controls which support data binding:

- asp:RadioButtonList
- asp:CheckBoxList
- asp:DropDownList
- asp:ListBox

The selectable items in each of the above controls are usually defined by one or more asp:ListItem controls, like this:

```
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="countrylist" runat="server">
<asp:ListItem value="N" text="Norway" />
<asp:ListItem value="S" text="Sweden" />
<asp:ListItem value="F" text="France" />
<asp:ListItem value="I" text="Italy" />
</asp:RadioButtonList>
</form>
</body>
</html>
```

However, with data binding we may use a separate source, like a database, an XML file, or a script to fill the list with selectable items.

By using an imported source, the data is separated from the HTML, and any changes to the items are made in the separate data source.

In the next three chapters, we will describe how to bind data from a scripted data source.

## ASP.NET - The ArrayList Object

**The ArrayList object is a collection of items containing a single data value.**

### Create an ArrayList

The ArrayList object is a collection of items containing a single data value.

Items are added to the ArrayList with the Add() method.

The following code creates a new ArrayList object named mycountries and four items are added:

```

<script runat="server">
Sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New ArrayList
    mycountries.Add( "Norway" )
    mycountries.Add( "Sweden" )
    mycountries.Add( "France" )
    mycountries.Add( "Italy" )
end if
end sub
</script>

```

By default, an `ArrayList` object contains 16 entries. An `ArrayList` can be sized to its final size with the `TrimToSize()` method:

```

<script runat="server">
Sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New ArrayList
    mycountries.Add( "Norway" )
    mycountries.Add( "Sweden" )
    mycountries.Add( "France" )
    mycountries.Add( "Italy" )
    mycountries.TrimToSize()
end if
end sub
</script>

```

An `ArrayList` can also be sorted alphabetically or numerically with the `Sort()` method:

```

<script runat="server">
Sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New ArrayList
    mycountries.Add( "Norway" )
    mycountries.Add( "Sweden" )
    mycountries.Add( "France" )
    mycountries.Add( "Italy" )
    mycountries.TrimToSize()
    mycountries.Sort()
end if
end sub
</script>

```

To sort in reverse order, apply the `Reverse()` method after the `Sort()` method:

```

<script runat="server">
Sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New ArrayList
    mycountries.Add( "Norway" )
    mycountries.Add( "Sweden" )
    mycountries.Add( "France" )
    mycountries.Add( "Italy" )
    mycountries.TrimToSize()
    mycountries.Sort()
    mycountries.Reverse()
end if
end sub
</script>

```

## Data Binding to an ArrayList

An ArrayList object may automatically generate the text and values to the following controls:

- asp:RadioButtonList
- asp:CheckBoxList
- asp:DropDownList
- asp:Listbox

To bind data to a RadioButtonList control, first create a RadioButtonList control (without any asp:ListItem elements) in an .aspx page:

```
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server" />
</form>
</body>
</html>
```

Then add the script that builds the list and binds the values in the list to the RadioButtonList control:

```
<script runat="server">
Sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New ArrayList
    mycountries.Add("Norway")
    mycountries.Add("Sweden")
    mycountries.Add("France")
    mycountries.Add("Italy")
    mycountries.TrimToSize()
    mycountries.Sort()
    rb.DataSource=mycountries
    rb.DataBind()
end if
end sub
</script>
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server" />
</form>
</body>
</html>
```

The DataSource property of the RadioButtonList control is set to the ArrayList and it defines the data source of the RadioButtonList control. The DataBind() method of the RadioButtonList control binds the data source with the RadioButtonList control.

**Note:** The data values are used as both the Text and Value properties for the control. To add Values that are different from the Text, use either the Hashtable object or the SortedList object.

# ASP.NET - The Hashtable Object

The Hashtable object contains items in key/value pairs.

## Create a Hashtable

The Hashtable object contains items in key/value pairs. The keys are used as indexes, and very quick searches can be made for values by searching through their keys.

Items are added to the Hashtable with the Add() method.

The following code creates a Hashtable named mycountries and four elements are added:

```
<script runat="server">
Sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New Hashtable
    mycountries.Add("N", "Norway")
    mycountries.Add("S", "Sweden")
    mycountries.Add("F", "France")
    mycountries.Add("I", "Italy")
end if
end sub
</script>
```

## Data Binding

A Hashtable object may automatically generate the text and values to the following controls:

- asp:RadioButtonList
- asp:CheckBoxList
- asp:DropDownList
- asp:Listbox

To bind data to a RadioButtonList control, first create a RadioButtonList control (without any asp:ListItem elements) in an .aspx page:

```
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server"
AutoPostBack="True" />
</form>
</body>
</html>
```

Then add the script that builds the list:

```
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New Hashtable
    mycountries.Add("N", "Norway")
    mycountries.Add("S", "Sweden")
    mycountries.Add("F", "France")
    mycountries.Add("I", "Italy")
    rb.DataSource=mycountries
    rb.DataValueField="Key"
```

```

rb.DataTextField="Value"
rb.DataBind()
end if
end sub
</script>
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server"
AutoPostBack="True" />
</form>
</body>
</html>

```

Then we add a sub routine to be executed when the user clicks on an item in the RadioButtonList control. When a radio button is clicked, a text will appear in a label:

```

<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New Hashtable
    mycountries.Add( "N" , "Norway" )
    mycountries.Add( "S" , "Sweden" )
    mycountries.Add( "F" , "France" )
    mycountries.Add( "I" , "Italy" )
    rb.DataSource=mycountries
    rb.DataValueField="Key"
    rb.DataTextField="Value"
    rb.DataBind()
end if
end sub
sub displayMessage(s as Object,e As EventArgs)
lbl1.text="Your favorite country is: " & rb.SelectedItem.Text
end sub
</script>
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server"
AutoPostBack="True" onSelectedIndexChanged="displayMessage" />
<p><asp:label id="lbl1" runat="server" /></p>
</form>
</body>
</html>

```

**Note:** You cannot choose the sort order of the items added to the Hashtable. To sort items alphabetically or numerically, use the SortedList object.

## ASP.NET - The SortedList Object

The **SortedList** object combines the features of both the **ArrayList** object and the **Hashtable** object.

### The SortedList Object

The **SortedList** object contains items in key/value pairs. A **SortedList** object automatically sort the items in alphabetic or numeric order.

Items are added to the **SortedList** with the **Add()** method. A **SortedList** can be sized to its final size with the **TrimToSize()** method.

The following code creates a SortedList named mycountries and four elements are added:

```
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New SortedList
    mycountries.Add("N", "Norway")
    mycountries.Add("S", "Sweden")
    mycountries.Add("F", "France")
    mycountries.Add("I", "Italy")
end if
end sub
</script>
```

## Data Binding

A SortedList object may automatically generate the text and values to the following controls:

- asp:RadioButtonList
- asp:CheckBoxList
- asp:DropDownList
- asp:Listbox

To bind data to a RadioButtonList control, first create a RadioButtonList control (without any asp:ListItem elements) in an .aspx page:

```
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server"
AutoPostBack="True" />
</form>
</body>
</html>
```

Then add the script that builds the list:

```
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New SortedList
    mycountries.Add("N", "Norway")
    mycountries.Add("S", "Sweden")
    mycountries.Add("F", "France")
    mycountries.Add("I", "Italy")
    rb.DataSource=mycountries
    rb.DataValueField="Key"
    rb.DataTextField="Value"
    rb.DataBind()
end if
end sub
</script>
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server"
AutoPostBack="True" />
</form>
</body>
</html>
```

Then we add a sub routine to be executed when the user clicks on an item in the RadioButtonList control. When a radio button is clicked, a text will appear in a label:

```
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New SortedList
    mycountries.Add("N", "Norway")
    mycountries.Add("S", "Sweden")
    mycountries.Add("F", "France")
    mycountries.Add("I", "Italy")
    rb.DataSource=mycountries
    rb.DataValueField="Key"
    rb.DataTextField="Value"
    rb.DataBind()
end if
end sub
sub displayMessage(s as Object,e As EventArgs)
lbl1.text="Your favorite country is: " & rb.SelectedItem.Text
end sub
</script>
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server"
AutoPostBack="True" onSelectedIndexChanged="displayMessage" />
<p><asp:label id="lbl1" runat="server" /></p>
</form>
</body>
</html>
```

## ASP .NET - XML Files

We can bind an XML file to a list control.

### An XML File

Here is an XML file named "countries.xml":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<countries>
<country>
<text>Norway</text>
<value>N</value>
</country>
<country>
<text>Sweden</text>
<value>S</value>
</country>
<country>
<text>France</text>
<value>F</value>
</country>
<country>
<text>Italy</text>
<value>I</value>
</country>
</countries>
```

## Bind a DataSet to a List Control

First, import the "System.Data" namespace. We need this namespace to work with DataSet objects. Include the following directive at the top of an .aspx page:

```
<%@ Import Namespace="System.Data" %>
```

Next, create a DataSet for the XML file and load the XML file into the DataSet when the page is first loaded:

```
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New DataSet
    mycountries.ReadXml(MapPath("countries.xml"))
end if
end sub
```

To bind the DataSet to a RadioButtonList control, first create a RadioButtonList control (without any asp:ListItem elements) in an .aspx page:

```
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server"
AutoPostBack="True" />
</form>
</body>
</html>
```

Then add the script that builds the XML DataSet:

```
<%@ Import Namespace="System.Data" %>
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New DataSet
    mycountries.ReadXml(MapPath("countries.xml"))
    rb.DataSource=mycountries
    rb.DataValueField="value"
    rb.DataTextField="text"
    rb.DataBind()
end if
end sub
</script>
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server"
AutoPostBack="True" onSelectedIndexChanged="displayMessage" />
</form>
</body>
</html>
```

Then we add a sub routine to be executed when the user clicks on an item in the RadioButtonList control. When a radio button is clicked, a text will appear in a label:

```
<%@ Import Namespace="System.Data" %>
<script runat="server">
```

```

sub Page_Load
if Not Page.IsPostBack then
    dim mycountries=New DataSet
    mycountries.ReadXml(MapPath("countries.xml"))
    rb.DataSource=mycountries
    rb.DataValueField="value"
    rb.DataTextField="text"
    rb.DataBind()
end if
end sub
sub displayMessage(s as Object,e As EventArgs)
lbl1.text="Your favorite country is: " & rb.SelectedItem.Text
end sub
</script>
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="rb" runat="server"
AutoPostBack="True" onSelectedIndexChanged="displayMessage" />
<p><asp:label id="lbl1" runat="server" /></p>
</form>
</body>
</html>

```

## ASP.NET - The Repeater Control

The Repeater control is used to display a repeated list of items that are bound to the control.

### Bind a DataSet to a Repeater Control

The Repeater control is used to display a repeated list of items that are bound to the control. The Repeater control may be bound to a database table, an XML file, or another list of items. Here we will show how to bind an XML file to a Repeater control.

We will use the following XML file in our examples ("cdcatalog.xml"):

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>
<year>1985</year>
</cd>
<cd>
<title>Hide your heart</title>
<artist>Bonnie Tyler</artist>
<country>UK</country>
<company>CBS Records</company>
<price>9.90</price>
<year>1988</year>
</cd>
<cd>
<title>Greatest Hits</title>
<artist>Dolly Parton</artist>
<country>USA</country>
<company>RCA</company>
<price>9.90</price>

```

```

<year>1982</year>
</cd>
<cd>
<title>Still got the blues</title>
<artist>Gary Moore</artist>
<country>UK</country>
<company>Virgin records</company>
<price>10.20</price>
<year>1990</year>
</cd>
<cd>
<title>Eros</title>
<artist>Eros Ramazzotti</artist>
<country>EU</country>
<company>BMG</company>
<price>9.90</price>
<year>1997</year>
</cd>
</catalog>

```

First, import the "System.Data" namespace. We need this namespace to work with DataSet objects. Include the following directive at the top of an .aspx page:

```
<%@ Import Namespace="System.Data" %>
```

Next, create a DataSet for the XML file and load the XML file into the DataSet when the page is first loaded:

```

<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycdcatalog=New DataSet
    mycdcatalog.ReadXml(MapPath( "cdcatalog.xml" ))
end if
end sub

```

Then we create a Repeater control in an .aspx page. The contents of the <HeaderTemplate> element are rendered first and only once within the output, then the contents of the <ItemTemplate> element are repeated for each "record" in the DataSet, and last, the contents of the <FooterTemplate> element are rendered once within the output:

```

<html>
<body>
<form runat="server">
<asp:Repeater id="cdcatalog" runat="server">
<HeaderTemplate>
...
</HeaderTemplate>
<ItemTemplate>
...
</ItemTemplate>
<FooterTemplate>
...
</FooterTemplate>
</asp:Repeater>
</form>
</body>
</html>

```

Then we add the script that creates the DataSet and binds the mycdcatalog DataSet to the Repeater control. We also fill the Repeater control with HTML tags and bind the data items to the cells in the <ItemTemplate> section with the <%#Container.DataItem("fieldname")%> method:

```
<%@ Import Namespace="System.Data" %>
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycdcatalog=New DataSet
    mycdcatalog.ReadXml(MapPath("cdcatalog.xml"))
    cdcatalog.DataSource=mycdcatalog
    cdcatalog.DataBind()
end if
end sub
</script>
<html>
<body>
<form runat="server">
<asp:Repeater id="cdcatalog" runat="server">
<HeaderTemplate>
<table border="1" width="100%">
<tr>
<th>Title</th>
<th>Artist</th>
<th>Country</th>
<th>Company</th>
<th>Price</th>
<th>Year</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td><%#Container.DataItem("title")%></td>
<td><%#Container.DataItem("artist")%></td>
<td><%#Container.DataItem("country")%></td>
<td><%#Container.DataItem("company")%></td>
<td><%#Container.DataItem("price")%></td>
<td><%#Container.DataItem("year")%></td>
</tr>
</ItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>
</form>
</body>
</html>
```

## Using the <AlternatingItemTemplate>

You can add an <AlternatingItemTemplate> element after the <ItemTemplate> element to describe the appearance of alternating rows of output. In the following example each other row in the table will be displayed in a light grey color:

```
<%@ Import Namespace="System.Data" %>
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycdcatalog=New DataSet
    mycdcatalog.ReadXml(MapPath("cdcatalog.xml"))
    cdcatalog.DataSource=mycdcatalog
    cdcatalog.DataBind()
```

```

end if
end sub
</script>
<html>
<body>
<form runat="server">
<asp:Repeater id="cdcatalog" runat="server">
<HeaderTemplate>
<table border="1" width="100%">
<tr>
<th>Title</th>
<th>Artist</th>
<th>Country</th>
<th>Company</th>
<th>Price</th>
<th>Year</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td><%#Container.DataItem("title")%></td>
<td><%#Container.DataItem("artist")%></td>
<td><%#Container.DataItem("country")%></td>
<td><%#Container.DataItem("company")%></td>
<td><%#Container.DataItem("price")%></td>
<td><%#Container.DataItem("year")%></td>
</tr>
</ItemTemplate>
<AlternatingItemTemplate>
<tr bgcolor="#e8e8e8">
<td><%#Container.DataItem("title")%></td>
<td><%#Container.DataItem("artist")%></td>
<td><%#Container.DataItem("country")%></td>
<td><%#Container.DataItem("company")%></td>
<td><%#Container.DataItem("price")%></td>
<td><%#Container.DataItem("year")%></td>
</tr>
</AlternatingItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>
</form>
</body>
</html>

```

## Using the <SeparatorTemplate>

The <SeparatorTemplate> element can be used to describe a separator between each record. The following example inserts a horizontal line between each table row:

```

<%@ Import Namespace="System.Data" %>
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycdcatalog=New DataSet
    mycdcatalog.ReadXml(MapPath("cdcatalog.xml"))
    cdcatalog.DataSource=mycdcatalog
    cdcatalog.DataBind()
end if
end sub
</script>

```

```

<html>
<body>
<form runat="server">
<asp:Repeater id="cdcatalog" runat="server">
<HeaderTemplate>
<table border="0" width="100%">
<tr>
<th>Title</th>
<th>Artist</th>
<th>Country</th>
<th>Company</th>
<th>Price</th>
<th>Year</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td><%#Container.DataItem("title")%></td>
<td><%#Container.DataItem("artist")%></td>
<td><%#Container.DataItem("country")%></td>
<td><%#Container.DataItem("company")%></td>
<td><%#Container.DataItem("price")%></td>
<td><%#Container.DataItem("year")%></td>
</tr>
</ItemTemplate>
<SeparatorTemplate>
<tr>
<td colspan="6"><hr /></td>
</tr>
</SeparatorTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>
</form>
</body>
</html>

```

## ASP.NET - The DataList Control

The DataList control is, like the Repeater control, used to display a repeated list of items that are bound to the control. However, the DataList control adds a table around the data items by default.

### Bind a DataSet to a DataList Control

The DataList control is, like the Repeater control, used to display a repeated list of items that are bound to the control. However, the DataList control adds a table around the data items by default. The DataList control may be bound to a database table, an XML file, or another list of items. Here we will show how to bind an XML file to a DataList control.

We will use the following XML file in our examples ("cdcatalog.xml"):

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>

```

```

<year>1985</year>
</cd>
<cd>
<title>Hide your heart</title>
<artist>Bonnie Tyler</artist>
<country>UK</country>
<company>CBS Records</company>
<price>9.90</price>
<year>1988</year>
</cd>
<cd>
<title>Greatest Hits</title>
<artist>Dolly Parton</artist>
<country>USA</country>
<company>RCA</company>
<price>9.90</price>
<year>1982</year>
</cd>
<cd>
<title>Still got the blues</title>
<artist>Gary Moore</artist>
<country>UK</country>
<company>Virgin records</company>
<price>10.20</price>
<year>1990</year>
</cd>
<cd>
<title>Eros</title>
<artist>Eros Ramazzotti</artist>
<country>EU</country>
<company>BMG</company>
<price>9.90</price>
<year>1997</year>
</cd>
</catalog>

```

First, import the "System.Data" namespace. We need this namespace to work with DataSet objects. Include the following directive at the top of an .aspx page:

```
<%@ Import Namespace="System.Data" %>
```

Next, create a DataSet for the XML file and load the XML file into the DataSet when the page is first loaded:

```

<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycdcatalog=New DataSet
    mycdcatalog.ReadXml(MapPath( "cdcatalog.xml" ))
end if
end sub

```

Then we create a DataList in an .aspx page. The contents of the <HeaderTemplate> element are rendered first and only once within the output, then the contents of the <ItemTemplate> element are repeated for each "record" in the DataSet, and last, the contents of the <FooterTemplate> element are rendered once within the output:

```

<html>
<body>
<form runat="server">
<asp:DataList id="cdcatalog" runat="server">

```

```

<HeaderTemplate>
...
</HeaderTemplate>
<ItemTemplate>
...
</ItemTemplate>
<FooterTemplate>
...
</FooterTemplate>
</asp:DataList>
</form>
</body>
</html>

```

Then we add the script that creates the DataSet and binds the mycdcatalog DataSet to the DataList control. We also fill the DataList control with a <HeaderTemplate> that contains the header of the table, an <ItemTemplate> that contains the data items to display, and a <FooterTemplate> that contains a text. Note that the gridlines attribute of the DataList is set to "both" to display table borders:

```

<%@ Import Namespace="System.Data" %>
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycdcatalog=New DataSet
    mycdcatalog.ReadXml(MapPath( "cdcatalog.xml" ))
    cdcatalog.DataSource=mycdcatalog
    cdcatalog.DataBind()
end if
end sub
</script>
<html>
<body>
<form runat="server">
<asp:DataList id="cdcatalog"
gridlines="both" runat="server">
<HeaderTemplate>
My CD Catalog
</HeaderTemplate>
<ItemTemplate>
" <%#Container.DataItem( "title" )%>" of
<%#Container.DataItem( "artist" )%> -
$ <%#Container.DataItem( "price" )%>
</ItemTemplate>
<FooterTemplate>
Copyright Hege Refsnes
</FooterTemplate>
</asp:DataList>
</form>
</body>
</html>

```

## Using Styles

You can also add styles to the DataList control to make the output more fancy:

```

<%@ Import Namespace="System.Data" %>
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
    dim mycdcatalog=New DataSet
    mycdcatalog.ReadXml(MapPath( "cdcatalog.xml" ))

```

```

cdcatalog.DataSource=mycdcatalog
cdcatalog.DataBind()
end if
end sub
</script>
<html>
<body>
<form runat="server">
<asp:DataList id="cdcatalog"
runat="server"
cellpadding="2"
cellspacing="2"
borderstyle="inset"
backcolor="#e8e8e8"
width="100%"
headerstyle-font-name="Verdana"
headerstyle-font-size="12pt"
headerstyle-horizontalalign="center"
headerstyle-font-bold="true"
itemstyle-backcolor="#778899"
itemstyle-forecolor="#ffffff"
footerstyle-font-size="9pt"
footerstyle-font-italic="true">
<HeaderTemplate>
My CD Catalog
</HeaderTemplate>
<ItemTemplate>
"<%#Container.DataItem("title")%>" of
<%#Container.DataItem("artist")%> -
$<%#Container.DataItem("price")%>
</ItemTemplate>
<FooterTemplate>
Copyright Hege Refsnes
</FooterTemplate>
</asp:DataList>
</form>
</body>
</html>

```

## Using the <AlternatingItemTemplate>

You can add an <AlternatingItemTemplate> element after the <ItemTemplate> element to describe the appearance of alternating rows of output. You may style the data in the <AlternatingItemTemplate> section within the DataList control:

```

<%@ Import Namespace="System.Data" %>
<script runat="server">
sub Page_Load
if Not Page.IsPostBack then
dim mycdcatalog=New DataSet
mycdcatalog.ReadXml(MapPath("cdcatalog.xml"))
cdcatalog.DataSource=mycdcatalog
cdcatalog.DataBind()
end if
end sub
</script>
<html>
<body>
<form runat="server">
<asp:DataList id="cdcatalog"
runat="server"
cellpadding="2"

```

```

cellspacing="2"
borderstyle="inset"
backcolor="#e8e8e8"
width="100%"
headerstyle-font-name="Verdana"
headerstyle-font-size="12pt"
headerstyle-horizontalalign="center"
headerstyle-font-bold="True"
itemstyle-backcolor="#778899"
itemstyle-forecolor="#ffffff"
alternatingitemstyle-backcolor="#e8e8e8"
alternatingitemstyle-forecolor="#000000"
footerstyle-font-size="9pt"
footerstyle-font-italic="True">
<HeaderTemplate>
My CD Catalog
</HeaderTemplate>
<ItemTemplate>
"<%#Container.DataItem("title")%>" of
<%#Container.DataItem("artist")%> -
$<%#Container.DataItem("price")%>
</ItemTemplate>
<AlternatingItemTemplate>
"<%#Container.DataItem("title")%>" of
<%#Container.DataItem("artist")%> -
$<%#Container.DataItem("price")%>
</AlternatingItemTemplate>
<FooterTemplate>
&copy; Hege Refsnes
</FooterTemplate>
</asp:DataList>
</form>
</body>
</html>

```

## ASP.NET - Database Connection

**ADO.NET is also a part of the .NET Framework. ADO.NET is used to handle data access. With ADO.NET you can work with databases.**

### What is ADO.NET?

- ADO.NET is a part of the .NET Framework
- ADO.NET consists of a set of classes used to handle data access
- ADO.NET is entirely based on XML
- ADO.NET has, unlike ADO, no Recordset object

### Create a Database Connection

We are going to use the Northwind database in our examples.

First, import the "System.Data.OleDb" namespace. We need this namespace to work with Microsoft Access and other OLE DB database providers. We will create the connection to the database in the Page\_Load subroutine. We create a dbconn variable as a new OleDbConnection class with a connection string which identifies the OLE DB provider and the location of the database. Then we open the database connection:

```

<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn

```

```

dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
end sub
</script>

```

**Note:** The connection string must be a continuous string without a line break!

## Create a Database Command

To specify the records to retrieve from the database, we will create a dbcomm variable as a new OleDbCommand class. The OleDbCommand class is for issuing SQL queries against database tables:

```

<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn,sql,dbcomm
dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
sql="SELECT * FROM customers"
dbcomm=New OleDbCommand(sql,dbconn)
end sub
</script>

```

## Create a DataReader

The OleDbDataReader class is used to read a stream of records from a data source. A DataReader is created by calling the ExecuteReader method of the OleDbCommand object:

```

<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn,sql,dbcomm,dbread
dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
sql="SELECT * FROM customers"
dbcomm=New OleDbCommand(sql,dbconn)
dbread=dbcomm.ExecuteReader()
end sub
</script>

```

## Bind to a Repeater Control

Then we bind the DataReader to a Repeater control:

```

<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">
sub Page_Load
dim dbconn,sql,dbcomm,dbread
dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
data source=" & server.mappath("northwind.mdb"))
dbconn.Open()
sql="SELECT * FROM customers"
dbcomm=New OleDbCommand(sql,dbconn)
dbread=dbcomm.ExecuteReader()
customers.DataSource=dbread

```

```

customers.DataBind()
dbread.Close()
dbconn.Close()
end sub
</script>
<html>
<body>
<form runat="server">
<asp:Repeater id="customers" runat="server">
<HeaderTemplate>
<table border="1" width="100%">
<tr>
<th>Companyname</th>
<th>Contactname</th>
<th>Address</th>
<th>City</th>
</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td><%#Container.DataItem("companyname")%></td>
<td><%#Container.DataItem("contactname")%></td>
<td><%#Container.DataItem("address")%></td>
<td><%#Container.DataItem("city")%></td>
</tr>
</ItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>
</form>
</body>
</html>

```

## **Close the Database Connection**

Always close both the DataReader and database connection after access to the database is no longer required:

```

dbread.Close()
dbconn.Close()

```

## **HTML Server Controls**

**HTML server controls are HTML tags understood by the server.**

### **HTML Server Controls**

HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a runat="server" attribute to the HTML element. This attribute indicates that the element should be treated as a server control.

**Note:** All HTML server controls must be within a <form> tag with the runat="server" attribute!

**Note:** ASP.NET requires that all HTML elements must be properly closed and properly nested.

<b>HTML Server Control</b>	<b>Description</b>
<u>HtmlAnchor</u>	Controls an <a> HTML element
<u>HtmlButton</u>	Controls a <button> HTML element
<u>HtmlForm</u>	Controls a <form> HTML element
<u>HtmlGeneric</u>	Controls other HTML element not specified by a specific HTML server control, like <body>, <div>, <span>, etc.
<u>HtmlImage</u>	Controls an <image> HTML element
<u>HtmlInputButton</u>	Controls <input type="button">, <input type="submit">, and <input type="reset"> HTML elements
<u>HtmlInputCheckBox</u>	Controls an <input type="checkbox"> HTML element
<u>HtmlInputFile</u>	Controls an <input type="file"> HTML element
<u>HtmlInputHidden</u>	Controls an <input type="hidden"> HTML element
<u>HtmlInputImage</u>	Controls an <input type="image"> HTML element
<u>HtmlInputRadioButton</u>	Controls an <input type="radio"> HTML element
<u>HtmlInputText</u>	Controls <input type="text"> and <input type="password"> HTML elements
<u>HtmlSelect</u>	Controls a <select> HTML element
<u>HtmlTable</u>	Controls a <table> HTML element
<u>HtmlTableCell</u>	Controls <td> and <th> HTML elements
<u>HtmlTableRow</u>	Controls a <tr> HTML element
<u>HtmlTextArea</u>	Controls a <textarea> HTML element

## Web Server Controls

Web server controls are special ASP.NET tags understood by the server.

### Web Server Controls

Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.

The syntax for creating a Web server control is:

```
<asp:control_name id="some_id" runat="server" />
```

<b>Web Server Control</b>	<b>Description</b>
<u>AdRotator</u>	Displays a sequence of images
<u>Button</u>	Displays a push button
<u>Calendar</u>	Displays a calendar
<u>CheckBox</u>	Displays a check box
<u>CheckBoxList</u>	Creates a multi-selection check box group
<u>GridView</u>	Displays fields of a data source in a grid
<u>DataSource</u>	Displays items from a data source by using templates
<u>DropDownList</u>	Creates a drop-down list
<u>HyperLink</u>	Creates a hyperlink
<u>Image</u>	Displays an image
<u>ImageButton</u>	Displays a clickable image
<u>Label</u>	Displays static content which is programmable (lets you apply styles to its content)
<u>LinkButton</u>	Creates a hyperlink button

<u>ListBox</u>	Creates a single- or multi-selection drop-down list
<u>Literal</u>	Displays static content which is programmable (does not let you apply styles to its content)
<u>Panel</u>	Provides a container for other controls
<u>PlaceHolder</u>	Reserves space for controls added by code
<u>RadioButton</u>	Creates a radio button
<u>RadioButtonList</u>	Creates a group of radio buttons
<u>Repeater</u>	Displays a repeated list of items bound to the control
<u>Table</u>	Creates a table
<u>TableCell</u>	Creates a table cell
<u>TableRow</u>	Creates a table row
<u>TextBox</u>	Creates a text box
<u>Xml</u>	Displays an XML file or the results of an XSL transform

## Validation Server Controls

**Validation server controls are used to validate user-input.**

### Validation Server Controls

A Validation server control is used to validate the data of an input control. If the data does not pass validation, it will display an error message to the user.

The syntax for creating a Validation server control is:

```
<asp:control_name id="some_id" runat="server" />
```

<b>Validation Server Control</b>	<b>Description</b>
<u>CompareValidator</u>	Compares the value of one input control to the value of another input control or to a fixed value
<u>CustomValidator</u>	Allows you to write a method to handle the validation of the value entered
<u>RangeValidator</u>	Checks that the user enters a value that falls between two values
<u>RegularExpressionValidator</u>	Ensures that the value of an input control matches a specified pattern
<u>RequiredFieldValidator</u>	Makes an input control a required field
<u>ValidationSummary</u>	Displays a report of all validation errors occurred in a Web page

# Web Services

## What are Web Services?

- Web services are application components
- Web services communicate using open protocols
- Web services are self-contained and self-describing
- Web services can be discovered using UDDI
- Web services can be used by other applications
- XML is the basis for Web services

## How does it Work?

**The basic Web services platform is XML + HTTP.**

The HTTP protocol is the most used Internet protocol.

XML provides a language, which can be used between different platforms, and programming languages and still express complex messages and functions.

## Web services platform elements

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

We will explain these topics later in the tutorial

## The Future of Web services

Don't expect too much, too soon.

The Web Services platform is a simple, interoperable, messaging framework. It still misses many important features like security and routing. But, these pieces will come once SOAP becomes more advanced.

Hopefully, Web services can make it much easier for applications to communicate.

## Why Web Services?

**A few years ago Web services were not fast enough to be interesting.**

Thanks to the major IT development the last few years, most people and companies have broadband connection and use the web more and more.

## Interoperability has highest priority.

When all major platforms could access the Web using Web browsers, different platforms could interact. For these platforms to work together, Web applications were developed.

Web applications are simple applications run on the web. These are built around the Web browser standards and can mostly be used by any browser on any platform.

## Web services take Web applications to the next level.

Using Web services your application can publish its function or message to the rest of the world.

Web services uses XML to code and decode your data and SOAP to transport it using open protocols.

With Web services your accounting departments Win 2k servers billing system can connect with your IT suppliers UNIX server.

### **Web services have two types of uses.**

#### **Reusable application components**

There are things different applications needs very often. So why make these over and over again?

Web services can offer application components like currency conversion, weather reports or even language translation as services.

Ideally, there will only be one type of each application component, and anyone can use it in their application.

#### **Connect existing software**

Web services help solve the interoperability problem by giving different applications a way to link their data.

Using Web services you can exchange data between different applications and different platforms.

## **Web Services Platform Elements**

#### **Web Services have three basic platform elements.**

**These are called SOAP, WSDL and UDDI.**

#### **What is SOAP?**

**The basic Web services platform is XML plus HTTP.**

- SOAP stands for Simple Object Access Protocol
- SOAP is a communication protocol
- SOAP is for communication between applications
- SOAP is a format for sending messages
- SOAP is designed to communicate via Internet
- SOAP is platform independent
- SOAP is language independent
- SOAP is based on XML
- SOAP is simple and extensible
- SOAP allows you to get around firewalls
- SOAP will be developed as a W3C standard

#### **What is WSDL?**

**WSDL is an XML-based language for describing Web services and how to access them.**

- WSDL stands for Web Services Description Language
- WSDL is written in XML
- WSDL is an XML document
- WSDL is used to describe Web services
- WSDL is also used to locate Web services
- WSDL is not yet a W3C standard

## What is UDDI?

**UDDI is a directory service where businesses can register and search for Web services.**

- UDDI stands for Universal Description, Discovery and Integration
- UDDI is a directory for storing information about web services
- UDDI is a directory of web service interfaces described by WSDL
- UDDI communicates via SOAP
- UDDI is built into the Microsoft .NET platform

## Web Service Example

Any application can have a Web Service component.

Web Services can be created regardless of programming language.

An example ASP.NET Web Service

In this example we use ASP.NET to create a simple Web Service.

```
<%@ WebService Language="VB" Class="TempConvert" %>

Imports System
Imports System.Web.Services

Public Class TempConvert :Inherits WebService

<WebMethod()> Public Function FahrenheitToCelsius
(ByVal Fahrenheit As Int16) As Int16
    Dim celsius As Int16
    celsius = (((Fahrenheit) - 32) / 9) * 5
    Return celsius
End Function

<WebMethod()> Public Function CelsiusToFahrenheit
(ByVal Celsius As Int16) As Int16
    Dim fahrenheit As Int16
    fahrenheit = (((Celsius) * 9) / 5) + 32
    Return fahrenheit
End Function
End Class
```

This document is a .asmx file. This is the ASP.NET file extension for XML Web Services.

**To run this example you will need a .NET server.**

The first line in this document that it is a Web Service, written in VB and the class name is "TempConvert":

```
<%@ WebService Language="VB" Class="TempConvert" %>
```

The next lines imports the namespace "System.Web.Services" from the .NET framework.

```
Imports System
Imports System.Web.Services
```

The next line defines that the "TempConvert" class is a WebService class type:

```
Public Class TempConvert :Inherits WebService
```

The next step is basic VB programming. This application has two functions. One to convert from Fahrenheit to Celsius, and one to convert from Celsius to Fahrenheit.

The only difference from a normal application is that this function is defined as a "WebMethod".

Use "WebMethod" to mark the functions in your application that you would like to make into web services.

```
<WebMethod()> Public Function FahrenheitToCelsius  
(ByVal Fahrenheit As Int16) As Int16  
    Dim celsius As Int16  
    celsius = (((Fahrenheit) - 32) / 9) * 5  
    Return celsius  
End Function  
  
<WebMethod()> Public Function CelsiusToFahrenheit  
(ByVal Celsius As Int16) As Int16  
    Dim fahrenheit As Int16  
    fahrenheit = (((Celsius) * 9) / 5) + 32  
    Return fahrenheit  
End Function
```

The last thing to do is to end the function and the class:

```
End Function  
  
End Class
```

If you save this as an .asmx file and publishes it on a server with .NET support, you should have your first working Web Service.

### **ASP.NET automates the process**

**With ASP.NET you do not have to write your own WSDL and SOAP documents.**

## **Web Service Use**

**These functions will send you a XML reply.**

These test use HTTP POST and will send a XML response like this:

```
<?xml version="1.0" encoding="utf-8" ?>  
<short xmlns="http://tempuri.org/">38</short>
```

### Use a form to access a Web Service.

Using a form and HTTP POST, you can put our web service on your site, like this:

Fahrenheit to Celsius:

Celsius to Fahrenheit:

### You can put our Web Service on your site.

Here is the code to put our Web Service on your site:

```
<form target="_blank" action='http://www.w3schools.com  
/webservices/tempconvert.asmx/FahrenheitToCelsius'  
method="POST">  
<table>  
    <tr>  
        <td>Fahrenheit to Celsius:</td>  
        <td><input class="frmInput" type="text"  
size="30" name="Fahrenheit"></td>  
    </tr>  
    <tr>  
        <td></td>  
        <td align="right"> <input type="submit"  
value="Submit" class="button"></td>  
    </tr>  
</table>  
</form>  
  
<form target="_blank" action='http://www.w3schools.com  
/webservices/tempconvert.asmx/CelsiusToFahrenheit'  
method="POST">  
<table>  
    <tr>  
        <td>Celsius to Fahrenheit:</td>  
        <td><input class="frmInput" type="text"  
size="30" name="Celsius"></td>  
    </tr>  
    <tr>  
        <td></td>  
        <td align="right"> <input type="submit"  
value="Submit" class="button"></td>  
    </tr>  
</table>  
</form>
```

**Web Services Summary:** This tutorial has taught you how to convert your applications into web-applications. You have learned how to use XML to send messages between applications. You have also learned how to export a function (create a web service) from your application.

# Apache

## What is Apache ?

In early 1995, a group of webmasters decided to get together and expand on the existing webserver software, and Apache was born - Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.

Today, Apache is the most used webserver on the Internet. Apache is Open Source freeware, and is available for Linux, Windows, and many versions of UNIX. The Apache Software Foundation (ASF) is a non-profit organization providing support for the Apache community of open-sourced software projects

## Apache HTTP Server

Apache HTTP Server is a project of the Apache Software foundation aimed at creating a robust, commercial-grade, featureful, and freely-available source code implementation of an HTTP (Web) server. Apache HTTP Server is a free software/open source web server for Unix-like systems, Microsoft Windows, Novell NetWare and other operating systems. Apache is notable for playing a key role in the initial growth of the World Wide Web, and continues to be the most popular web server in use serving as the de facto reference platform against which other web servers are designed and judged.

Apache features configurable error messages, DBMS-based authentication databases, and content negotiation. It is also supported by several graphical user interfaces (GUIs) which permit easier, more intuitive configuration of the server.

### **Features of Apache HTTP Server :**

It is a powerful, flexible, HTTP/1.1 compliant web server

It implements the latest protocols, including HTTP/1.1 -FC2616)

It is highly configurable and extensible with third-party modules

It can be customised by writing 'modules' using the Apache module API

It provides full source code and comes with an unrestrictive license

It runs on Windows 2003/XP/2000/NT/9x, Netware 5.x and above, OS/2, and most versions of Unix, as well as several other operating systems

It is actively being developed

It encourages user feedback through new ideas, bug reports and patches

## History

The first version of the Apache web server was created by Rob McCool, who was heavily involved with the National Center for Supercomputing Applications web server, known simply as NCSA HTTPd. When Rob left NCSA in mid-1994, the development of httpd stalled, leaving a variety of patches for improvements circulating through e-mails.

Rob McCool was not alone in his efforts. Several other developers helped form the original "Apache Group": Brian Behlendorf, Roy T. Fielding, Rob Hartill, David Robinson, Cliff Skolnick, Randy Terbush, Robert S. Thau, Andrew Wilson, Eric Hagberg, Frank Peters, and Nicolas Pioch.

The FAQ on the project's official site states: "The name 'Apache' was chosen from respect for the Native American Indian tribe of Apache (Indé), well-known for their superior skills in warfare strategy and their inexhaustible endurance." However, the most widespread interpretation is that

the name comes from the fact that when it was developed in early 1995, the web server consisted a set of patches to the codebase of NCSA HTTPd 1.3 and was therefore "a patchy" server. This was the explanation initially given on the project's website.

When first released, Apache was the only viable open source alternative to the Netscape web server (currently known as Sun Java System Web Server). It has since evolved to rival other Unix-based web servers in terms of functionality and performance. Since April 1996 Apache has been the most popular HTTP server on the Internet. By May 1999 Apache installations served 57% of all websites. Its popularity continued to rise, and in February 2006 Apache served 68% of all websites.[3] Microsoft's Internet Information Services (IIS) is the main competitor to Apache, trailed by Sun Microsystems' Sun Java System Web Server and a host of other applications such as Zeus.

## License

The License under which software from the Apache Foundation is distributed is a distinctive part of the Apache HTTP Server's history and presence in the open source software environment. The Apache License allows for the distribution of both open- and closed-source derivations of the source code.

Furthermore, it is perhaps surprising that the Free Software Foundation does not consider the Apache License to be "compatible" with version 2.0 of the GNU General Public License (GPL), meaning that software licensed under the Apache License cannot be integrated with software that is distributed under the GPL. Here is what the FSF says about the Apache License. The current draft of Version 3 of the GPL includes a provision (Section 7e) which allows it to be compatible with licenses that have patent retaliation clauses, including the Apache License.

## Usage

Apache is primarily used to serve static and dynamic content on the World Wide Web. Many web applications are designed expecting the environment and features that Apache provides. Apache is the web server component of the popular LAMP web server application stack, alongside Linux, MySQL, and the PHP/Perl/Python programming languages.

Apache is redistributed as part of various proprietary packages, such as the Oracle database or the IBM WebSphere application server. Mac OS X integrates Apache as its built-in web server and as support for its WebObjects application server. It is also supported in some way by Borland in the Kylix and Delphi development tools. Apache is included with Novell NetWare 6.5, where it is the default web server.

Apache is used for many other tasks where content needs to be made available in a secure and reliable way. One example is sharing files from a personal computer over the Internet. A user who has Apache installed on their desktop can put arbitrary files in the Apache's document root which can then be shared. Programmers developing web applications often use a locally installed version of Apache in order to preview and test code as it is being developed.

## Features

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, Tcl, and PHP. Popular authentication modules include mod\_access, mod\_auth, and mod\_digest. A sample of other features include SSL and TLS support (mod\_ssl), a proxy module, a useful URL rewriter (also known as a rewrite engine, implemented under mod\_rewrite), custom log files (mod\_log\_config), and filtering support (mod\_include and mod\_ext\_filter). Apache logs can be analyzed through a web browser using free scripts such as AWStats or Visitors.

## Apache 2

Version 2 of the Apache server was a substantial re-write of much of the code, with a strong focus on further modularisation and the development of a portability layer; the APR. The Apache 2.x core has several major enhancements over Apache 1.x.[1] These include UNIX threading, better support for non-Unix platforms (such as Windows), a new Apache API, and IPv6 support. The first alpha release of Apache 2 was in March 2000 with the first general availability release in May 2002. Version 2.2 introduced a new auth API that allows for more flexibility. It also features improved cache modules and proxy modules

# Internet Information Services (IIS)

Internet Information Services (IIS) is the Windows 2000 Web service that makes it easy to publish information on your intranet for the Internet.

## To configure your server as an application server

1. Click **Start**, point to **Settings**, and click **Network and Dial-up Connections**.
  2. Select **Local Area Connection** and open its properties.
  3. Select **File and Printer Sharing for Microsoft Networks** and open its properties.
  4. On the **Server Optimization** tab, select **Maximize data throughput for network applications**.
- **Socket Pooling, Performance, and Security Issues**

You might want to disable socket pooling if any of the following are true:

- You are not hosting a large number of sites.
- You have special security concerns.

Socket pooling will cause IIS 5.0 to listen to all IP addresses, which might present a possible security risk for secure domains with multiple networks. In addition, both bandwidth throttling and performance adjustments will apply to all Web sites configured for the same port, for example port 80. If you intend to use bandwidth throttling or do performance tuning on a per-site basis, you will need to disable socket pooling.

To disable socket pooling, type the following at the command prompt:

```
c:\inetpub\adminscripts\cscript adsutil.vbs set w3svc/disablesocketpooling true
```

The command prompt will reply:

```
disablesocketpooling : (BOOLEAN) True
```

- **Stopping Internet Services**

The Reliable Restart feature of the Windows 2000 Service Control Manager will automatically restart Internet services if the Inetinfo.exe process terminates abnormally, or if you use Windows Task Manager or Kill.exe to stop Internet services. For more information, see Restarting IIS. If you want to stop Internet services, you must disable Reliable Restart.

## To disable Reliable Restart

0. Click **Start**, point to **Settings**, click **Control Panel**, double-click **Administrative Tools**, then double-click **Computer Management**.
1. Under the **System Tools** node, click **Services**.
2. Open the properties for **IIS Admin Service**, select the **Recovery** tab, and choose **Take No Action** in each of the drop-down menus.

You can also disable Reliable Restart by typing **Iisreset.exe /disable** at the command line.

- **ASP Buffering and Performance After Upgrade**

In IIS 4.0, the **Buffer** property of the **ASP Response** object was set to FALSE by default. In a new installation of IIS 5.0, the **Buffer** property is set to TRUE by default. During an upgrade to IIS 5.0, the **Buffer** property will not be changed from its previous setting.

Setting the **Buffer** property to TRUE can significantly improve the performance of large ASP applications in which users primarily connect to the application by means of a modem. You can enable buffering for your applications from the IIS snap-in or by adding the `<% Response.Buffer = True %>` statement to selected pages. You can also change the property for entire applications by using the IIS snap-in. For more information, see the IIS 5.0 documentation.

## Multiple Sites and Clustering

- **Remove All IIS 5.0 Resources from a Cluster Before Uninstalling Microsoft Cluster Service**

All IIS resources, including NNTP and SMTP resources, must be removed from a cluster before Microsoft® Clustering is uninstalled. If they are not removed, you will not be able to stop or start the previously clustered IIS sites.

To fix this, type the following at the command prompt for all previously clustered IIS resources:

```
Inetpub\AdminScripts\adsutil.vbs set <service name>/<instance id>/ClusterEnabled 0
```

- **FrontPage Server Extensions Not Supported in Microsoft Clustering**

Microsoft® FrontPage® Server Extensions are not supported for resources in Microsoft Clustering.

- **Stopping Internet Services in Microsoft Clustering**

When using the new **Restart IIS** option in the IIS snap-in, or when using `lisisreset.exe` (the command-line version of this feature), to stop IIS 5.0, Microsoft Clustering will attempt to automatically restart IIS.

### To keep Microsoft Clustering from automatically restarting IIS

1. Open a connection to the cluster where IIS 5.0 cluster resources are located and take all IIS cluster resources off line.
  2. In the IIS snap-in, right-click the Web service and click **Restart IIS**.
- **ASP and FTP Sessions Will Not Fail Over**

ASP and FTP sessions do not fail over to other nodes in Microsoft Clustering. ASP session information is not lost if the affected computer fails back before the session times out, and if the original failure wasn't due to IIS stopping unexpectedly. However, FTP sessions are lost and must be restarted. In both cases, clients will need to re-establish the connection if a failover occurs on the server.

- **Clustered NNTP and SMTP Resources Must be Stopped by Using Cluster Administrator User Interface**

NNTP and SMTP resources that are part of Microsoft Clustering must be stopped using the Cluster Administration user interface. Using the IIS snap-in will not stop them.

- **Using Clustered Host Header Sites**

If you are using clustered Web sites that use host headers, you must set the ServerAutoStart metabase property to TRUE for these sites.

To set this property type the following at the command prompt:

```
%SystemDrive%\inetpub\adminscripts\adsutil.vbs set w3svc<instance id>/ServerAutoStart True
```

where <instance id> is the instance ID of the virtual host sites that are part of the cluster.

- **Deleting Clustered Resources**

If you are deleting clustered IIS 5.0 resources, you need to use a two-step process.

1. In the Cluster Administration user interface, remove the resources from the cluster.
2. In the IIS snap-in, delete the resources.

**To upgrade to the FrontPage 2000 Server Extensions from the Default Web Site:**

1. Open **Add/Remove Programs** in Control Panel.
2. Click on **Add/Remove Windows Components** to bring up the **Windows Components Wizard**.
3. Click **Next**.
4. Select **Internet Information Services (IIS)** and click **Details**.
5. Select **FrontPage 2000 Server Extensions** and click **OK**.
6. Click **Next** to complete the Wizard and to install the FrontPage 2000 Server Extensions.

**To upgrade to the FrontPage 2000 Server Extensions on other virtual Web sites:**

7. At the command prompt, navigate to the \Program Files\Common Files\Microsoft Shared\Web Server Extensions\40\bin directory.
8. In this directory, type the following at the command prompt: **fpsrvadm -o upgrade -p all**

- **Problems Starting Out-of-Process Applications After Changing Account Information**

This is due to the user account's username and password information not being synchronized after the change. You might receive Event Log errors telling you that your IWAM\_computername account could not be logged on. If you encounter this problem, run the synciwam script to synchronize the passwords.

To run the script, at the command prompt type **script synciwam.vbs [-v|-h] -v** uses verbose mode and prints a log of the script's activity. **-h** prints the script Help information.

- **Using the MyInfo, PageCounter, and Counters Objects After an Upgrade**

To use your current data for these objects after an upgrade to IIS 5.0, you must move their data files to the %WINDIR%\inetsrv\Data directory, which is created by IIS during the upgrade. Move the MyInfo.xml and the Counters.txt files from the %WINDIR%\inetsrv\ directory and the %WINDIR%\HitCnt.cnt file to the new directory %WINDIR%\inetsrv\Data. ("%WINDIR%" is the Windows installation directory.) The components will then append information to your files.

## To set IIS so that users see the new IIS 5.0 welcome page

1. Open the **Properties Sheet** for the Default Web Site in the IIS snap-in.
2. Click the **Documents** tab.
3. Select the file **default.asp** and click **Remove**.

If you want visitors to your site to see a welcome page that you created, replace the IIS 4.0 Default.asp file with your own Default.asp file.

- **Hosting Multiple Web Sites**

The IIS 5.0 documentation says that if you use multiple IP addresses to host multiple Web sites, you will need an additional network card for each IP address. In fact, it is possible to bind multiple IP addresses to a single network card, although this configuration is not recommended for sites with high volumes of Internet traffic.

**Note** Microsoft® Windows 2000® Professional with IIS 5.0 can host one Web site and one FTP site on a single computer. If you would like to host multiple Web or FTP sites on a single computer, consider upgrading to Microsoft Windows 2000 Server.

- **Documentation Correction for DisableSocketPooling**

The IIS documentation states that sockets are shared between sites that use the same socket number but different IP addresses. This is incorrect. Sockets are shared between sites that use the same port number but different IP addresses.

- **Adsutil.exe Not Included in Samples**

The Adsutil.exe utility is not included in the samples, as mentioned in the documentation.

## Getting Help and Providing Feedback

Peer-to-peer newsgroups are available to help you interact with other users of our products. You can use any newsreader software to access these newsgroups, but you might need to configure it in order to read them. When prompted for News Server, specify **msnews.microsoft.com**. You do not need to enter an account name or password. Before posting to the newsgroups, please review the Microsoft Newsgroup Rules of Conduct.

For IIS 5.0 issues, please use: **microsoft.public.inetsrvr.iis**

## Installing IIS

Internet Information Services is installed on Windows 2000 Server by default. You can remove IIS or select additional components by using the Add/Remove Programs application in Control Panel.

## To install IIS, add components, or remove components

1. Click **Start**, point to **Settings**, click **Control Panel** and start the **Add/Remove Programs** application.
2. Select **Configure Windows**, click the **Components** button, and then follow the on-screen instructions to install, remove, or add components to IIS.

**Note** If you *upgraded* to Windows 2000, IIS 5.0 will be installed by default only if IIS was installed on your previous version of Windows.

**Tip** You can find unattended installation information in the *Deployment Planning Guide* volume of the Windows 2000 Server Resource Kit.

## Directories remaining after uninstall

The following directories containing user content will remain on your system after you completely uninstall IIS:

- \Inetpub
- \systemroot\Help\iisHelp
- \systemroot\system32\inetsrv

## Related Topics

For more information on related topics such as the IIS software checklist, security checklist, or troubleshooting, see the Windows documentation by clicking **Help** on the **Start** menu.

## Features

Internet Information Services 5.0 has many new features to help Web administrators to create scalable, flexible Web applications.

### Security ,Administration ,Programmability ,Internet Standards ,Security

- **Digest Authentication:** Digest authentication allows secure and robust authentication of users across proxy servers and firewalls. In addition, Anonymous, HTTP Basic, and integrated Windows authentication (formerly known as Windows NT Challenge/Response authentication and NTLM authentication) are still available.
- **Secure Communications:** Secure Sockets Layer (SSL) 3.0 and Transport Layer Security (TLS) provide a secure way to exchange information between clients and servers. In addition, SSL 3.0 and TLS provide a way for the server to verify who the client is *before* the user logs on to the server. In IIS 5.0, client certificates are exposed to both ISAPI and Active Server Pages, so that programmers can track users through their sites. Also, IIS 5.0 can map the client certificate to a Windows user account, so that administrators can control access to system resources based on the client certificate.
- **Server-Gated Cryptography:** Server-Gated Cryptography (SGC) is an extension of SSL that allows financial institutions with export versions of IIS to use strong 128-bit encryption. Although SGC capabilities are built into IIS 5.0, a special SGC certificate is required to use SGC.
- **Security Wizards:** Security wizards simplify server administration tasks.

The Web Server Certificate Wizard simplifies certificate administration tasks, such as creating certificate requests and managing the certificate life cycle.

The Permissions Wizard makes it easy to configure Web site access by assigning access policies to virtual directories and files. The Permissions Wizard can also update NTFS file permissions to reflect these Web access policies.

The CTL wizard helps you configure your certificate trust lists (CTLs). A CTL is a list of trusted certification authorities (CAs) for a particular directory. CTLs are especially useful for Internet service providers (ISPs) who have several Web sites on their server and who need to have a different list of approved certification authorities for each site.

- **IP and Internet Domain Restrictions:** You can grant or deny Web access to individual computers, groups of computers, or entire domains.
- **Kerberos v5 Authentication Protocol Compliance:** IIS is fully integrated with the Kerberos v5 authentication protocol implemented in Microsoft® Windows® 2000, allowing you to pass authentication credentials among connected computers running Windows.
- **Certificate Storage:** IIS certificate storage is now integrated with the Windows CryptoAPI storage. The Windows Certificate Manager provides a single point of entry that allows you to store, back up, and configure server certificates.
- **Fortezza:** The U.S. government security standard, commonly called Fortezza, is supported in IIS 5.0. This standard satisfies the Defense Message System security architecture with a cryptographic mechanism that provides message confidentiality, integrity, authentication, and access control to messages, components, and systems. These features can be implemented both with server and browser software and with PCMCIA card hardware.

## Administration

- **Restarting IIS:** Now you can restart your Internet services without having to reboot your computer.
- **Backing Up and Restoring IIS:** You can back up and save your metabase settings to make it easy to return to a safe, known state.
- **Process Accounting:** Provides information about how individual Web sites use CPU resources on the server. This information is useful in determining which sites are using disproportionately high CPU resources or which might have malfunctioning scripts or CGI processes.
- **Process Throttling:** You can limit the percentage of time the CPU spends processing out-of-process ASP, ISAPI, and CGI applications for individual Web sites. In addition, misbehaving processes can be stopped and restarted.
- **Improved Custom Error Messages:** Now administrators can send informative messages to clients when HTTP errors occur on their Web sites. Also includes detailed ASP error processing capabilities through the use of the 500-100.asp custom error message. You can use the custom errors that IIS 5.0 provides, or create your own.
- **Configuration Options:** You can set permissions for Read, Write, Execute, Script, and FrontPage Web operations at the site, directory, or file level.
- **Remote Administration:** IIS 5.0 has Web-based administration tools that allow remote management of your server from almost any browser on any platform. With IIS 5.0, you can set up administration accounts called Operators with limited administration privileges on Web sites, to help distribute administrative tasks.
- **Terminal Services:** Terminal Services is a feature of Windows 2000 that allows you to run 32-bit Windows applications on terminals and terminal emulators running on personal computers and other computer desktops. Terminal Services allows virtually any desktop to run applications on the server. This enables you to remotely administer Windows 2000 services such as IIS as if you were at the server console, including administration from older legacy PCs, or even non-PC devices such as UNIX workstations with compatible client software. (Non-Windows-based client devices require third-party add-on software.)

- **Centralized Administration:** Administration tools for IIS use the Microsoft® Management Console (MMC). MMC hosts the programs, called snap-ins, that administrators use to manage their servers. You can use IIS snap-in from a computer running Windows 2000 Professional to administer a computer on your intranet running Internet Information Services on Windows 2000 Server.

## Programmability

- **Active Server Pages:** You can create dynamic content by using server-side scripting and components to create browser-independent dynamic content. Active Server Pages (ASP) provides an easy-to-use alternative to CGI and ISAPI by allowing content developers to embed any scripting language or server component into their HTML pages. ASP provides access to all of the HTTP request and response streams, as well as standards-based database connectivity and the ability to customize content for different browsers.
- **New ASP Features:** Active Server Pages has some new and improved features for enhancing performance and streamlining your server-side scripts.
- **Application Protection:** IIS 5.0 offers greater protection and increased reliability for your Web applications. By default, IIS will run all of your applications in a common or *pooled* process that is separate from core IIS processes. In addition, you can still *isolate* mission-critical applications that should be run outside of both core IIS and pooled processes.
- **ADSI 2.0:** In IIS 5.0, administrators and application developers will have the ability to add custom objects, properties, and methods to the existing ADSI provider, giving administrators even more flexibility in configuring their sites.

## Internet Standards

- **Standards Based:** Microsoft Internet Information Services 5.0 complies with the HTTP 1.1 standard, including features such as PUT and DELETE, the ability to customize HTTP error messages, and support for custom HTTP headers.
  - **Multiple Sites, One IP Address:** With support for host headers, you can host multiple Web sites on a single computer running Microsoft Windows 2000 Server with only one IP address. This is useful for Internet service providers and corporate intranets hosting multiple sites.
  - **Web Distributed Authoring and Versioning (WebDAV):** Enables remote authors to create, move, or delete files, file properties, directories, and directory properties on your server over an HTTP connection.
  - **News and Mail:** You can use SMTP and NNTP Services to set up intranet mail and news services that work in conjunction with IIS.
  - **PICS Ratings:** You can apply Platform for Internet Content Selection (PICS) ratings to sites that contain content for mature audiences.
  - **FTP Restart:** Now File Transfer Protocol file downloads can be resumed without having to download the entire file over again if an interruption occurs during data transfer.
  - **HTTP Compression:** Provides faster transmission of pages between the Web server and compression-enabled clients. Compresses and caches static files, and performs on-demand compression of dynamically generated files.
-

## **Quick Site Setup with IIS**

IIS creates a default Web site and FTP site when you install Windows 2000 Server. This topic describes how to publish information on those default sites.

### **To publish content on your Web site**

1. Create a home page for your Web site. See Choosing an Authoring Tool for more information on available tools used in Web site creation.
2. Name your home page file Default.htm or Default.asp.
3. Copy your home page into the default Web publishing directory for IIS. The default Web publishing directory is also called the home directory, and the location provided by Setup is \Inetpub\Wwwroot.
4. If your network has a name resolution system (typically DNS), then visitors can simply type your computer name in the address bar of their browsers to reach your site. If your network does not have a name resolution system, then visitors must type the numerical IP address of your computer. For more information, see About Name Resolution.

### **To publish content on your FTP site**

1. Copy or move your files into the default FTP publishing directory. The default directory provided by Setup is \Inetpub\Ftproot.
2. If your network has a name resolution system (typically DNS), then visitors can type **ftp://** followed by your computer name in the address bar of their browsers to reach your site. If not, then visitors must type **ftp://** and the numerical IP address of your computer. To customize the appearance of your FTP site, see Setting FTP Messages and Directory Output Style.

## **FTP**

Data on most systems is represented in units called files. Data transfer between systems usually involves transfer of files. TCP/IP application service provides facilities for transferring files to and fro from remote computer systems. Usually the user transferring a file needs authority to login and access files on the remote system.

FTP stands for File Transfer Protocol and is a way to transfer large files, programs and images from a server to your computer. FTP can be used to transfer both binary and ASCII or text files reliably and at high speeds. FTP works on the client/server principle. Files that can be transferred are stored on computers called FTP servers. FTP software (like Fetch or WS\_FTP) that resides on your machine allows you to interact with a server elsewhere to access information (files) or to transfer files.

Browser like Netscape Navigator also allows what's known as anonymous FTP. This means that you can connect to an FTP archive, navigate to find the file you want then transfer the file to your hard drive without leaving your browser. Remember that you have to tell browser that you are using a different protocol - that you are using ftp rather than http. Just remember to type `ftp://` - then the address of the site. Most of the good stuff can be found in the public (pub) directories.

The most famous use of FTP is known as Anonymous FTP. This system is one of the oldest and most popular services available on the net. System operators who maintain Internet computers regard security as one of their highest priorities. Allowing only authorized access and preventing uninvited visitors to their sites is one of their primary responsibilities. Security is most often controlled through the use of personal accounts, login names, and passwords. This restricts use of the computer and access to the computer information to authorized users with registered accounts. This is in direct contrast to the notion of the Internet as a repository of information with essentially unlimited access.

Anonymous FTP is the primary solution to this dilemma.

The FTP server program is installed on an Internet computer, which creates a partition or separate section of the computers storage. The separation is so complete that a user allowed accessing the FTP partition is prevented from accessing any other part of the system. Normal login and password access is also designed to isolate users from sensitive portions of the computer but the FTP protection has proven to be essentially foolproof as witnessed by its almost universal use. The key feature of the FTP partition that allows such good security is that it can only be used for file transfer, naturally file transfer according to the FTP protocol. Therefore FTP partitions, normally referred to as FTP sites, are used to store information with universal access.

The most common method of access is through the process of Anonymous FTP. An Internet user who wants to access the information available through FTP uses an FTP client program to contact the FTP server running on an FTP site. The FTP process is similar to the Telnet process in that the user is expected to login to the FTP site. The big difference is that everyone is allowed to use the same login name. The universal login name is the word anonymous, hence the phrase Anonymous FTP. In years past, the accepted password was guest but this has been replaced with a request to the user to enter their e-mail address. Some systems still accept guest as a password but using the e-mail address gives the host system a chance to monitor use of the system. There are also some systems that use the username `ftp` or `guest` instead of `anonymous` but these are rare.

## **FTP Components:**

FTP is defined in RFC 959. FTP session normally involves the interaction of five software elements:

1. User Interface: This provides a user interface and drives the client Protocol Interpreter.
2. Client PI: This is the client protocol interpreter. It issues commands to the remote server protocol interpreter and it also drives the server data transfer process.
3. Server PI: This is the server protocol interpreter. It responds to commands issued by the client protocol interpreter and drives the server data transfer process.
4. Client DTP: This is the client data transfer process responsible for communicating with the server data transfer process and the local file system.
5. Server DTP:

6. This is the server data transfer process responsible for communicating with the client data transfer process and the remote file system.

During an FTP session there will be two separate network connections one between PIs, and one between the DTPs. The connection between the PIs is known as control connection and the connection between the DTPs is known as the data connection.

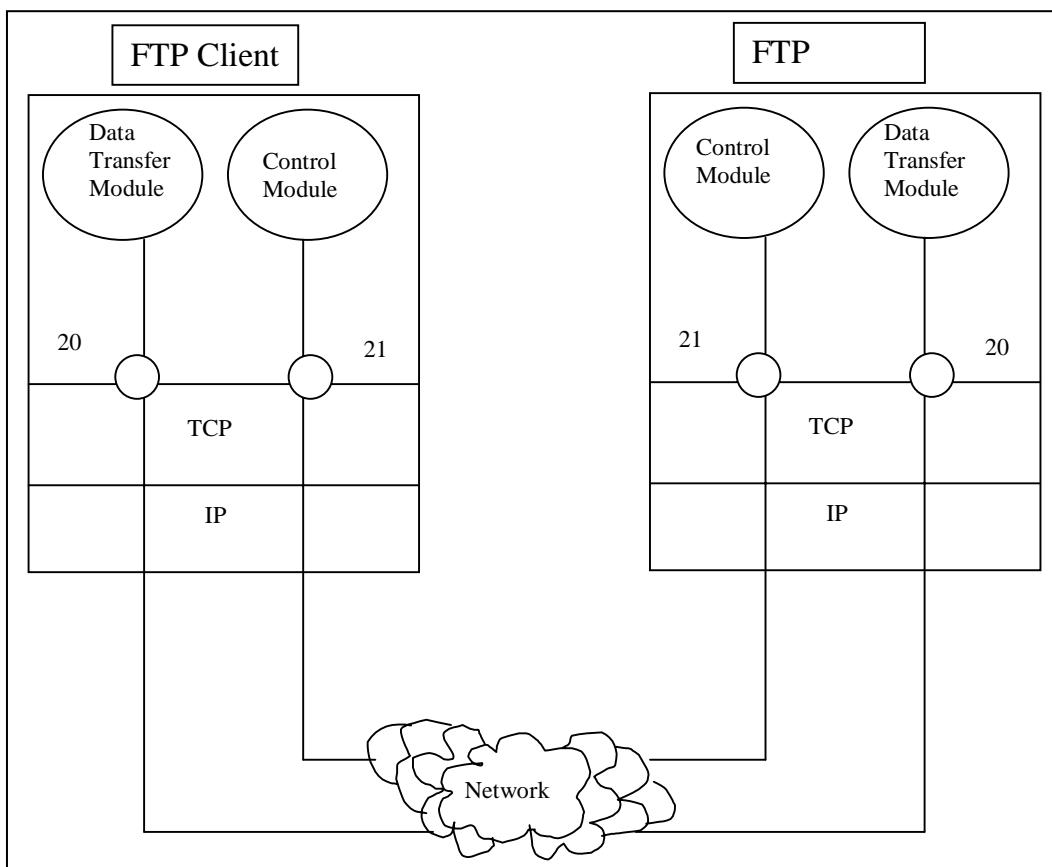
## **FTP Control Connection Establishment**

FTP Client software runs on user's machine and initiates connection to the FTP server. FTP server listens on the TCP port 21 and waits for connection requests from clients. When FTP server receives the connection request, a three-way handshake establishes a TCP connection. This connection is called Control connection (with port 21 of the FTP server). Control connection is used for sending FTP commands and responses.

## **FTP Data Transfer Connection establishment**

FTP Client issues a command to start a Data transfer. FTP Client starts a Data Transfer process by sending the local port number to the FTP server through the control connection already established. FTP server receives the port number of the FTP client for data transfer process. FTP server starts the Data Transfer process on the port no. 20, which issues a TCP connection request to connect to the data process port number on the FTP client. This connection is called Data transfer connection. Data transfer connection is used only for transferring the data from the requested files. FTP permits both Download and upload of files to the FTP server.

## FTP Operation Model



### Two connections established in an FTP session:

The use of separate connections for control and data offers advantages that the two connections can select different appropriate qualities of service e.g. minimum delay for the control connection and maximum throughput for the data connection. Data connection exists only for the duration of the file transfer and is closed after the data transfer is completed. New data connection is established every time a file is transferred. Control connection remains open

### FTP from a user perspective:

Users with registered accounts on Internet computers usually have FTP access to their personal accounts as well as to anonymous FTP sites. Accessing your personal account will require the normal username and password used during normal Telnet access. Using FTP to access your own personal account usually expands the access from just download to both upload and download, as well as directory creation/deletion, and file deletion. The amount of access depends on the sophistication of the FTP client software. FTP access to personal accounts is not intended to nor will it replace Telnet access; however, it is the best option for data transfer between multiple Internet accounts, or your Internet account and your desktop PC.

The kind and amount of information available through anonymous FTP is virtually limitless. Software programs, including the most popular Web Browsers, weather maps, government documents, etc., are all available at various FTP sites. Some of the activities are listed here:

- Listing of files in the remote directory.
- Renaming and deleting the files. (with permission)
- Transferring files from remote host to local host (downloading)

- Transferring files from local host to remote host ( uploading)

There are mainly two types of client TCP/IP applications available:

- Command line Interface
- Graphical User Interface

## Command Line Interface

To start an FTP session:

ftp [hostname] e.g. ftp internic.net

And if the IP address of the site is known then use

ftp 137.65.4.1

If the host is reachable you will see a "Welcome" message and then it prompts for username and password

Name: anonymous

Password: anonymous or your e-mail id

You will get an ftp prompt "ftp>" as below:

ftp>

Help Command:

ftp> ?

```
cd      bell    close   delete  debug
dir     get     help    ls       mkdir
open    pwd     quit   rmdir   send
type   trace
```

Available commands are listed (most of the commands are UNIX commands). You can use these commands to change directory, display the list of files in the directory, download/upload the files.

Meaning of some commands:

cd remote\_directory

Changes the working directory on the remote machine to remote\_directory.

delete remote\_file

Deletes the file remote\_file on the remote machine.

dir [ remote\_directory] [local\_file]

Prints a listing of the directory contents in the directory, remote\_directory, and, optionally, placing the output in local\_file.

disconnect

A synonym for close

get remote\_file

Retrieves the remote\_file and stores it on the local machine.

help [command]

Prints an informative message about the meaning of command.

lcd [directory]

Changes the working directory on the local machine. If no directory is specified the user's home directory is used.

ls

Prints an abbreviated listing of the contents of a directory on the remote machine.

### To copy a file from a FTP server to FTP client:

ftp> get <name\_of\_the\_remote\_file>

Remote file will be downloaded to the current directory in the FTP client.

ftp> get < name\_of\_the\_remote\_file > <name\_of\_the\_local\_file>

In the case remote file will be downloaded to destination file in the local host.

**Closing/exiting the FTP connection:**

```
ftp> close  
goodbye  
ftp>
```

**To exit ftp session completely:**

```
ftp>bye  
%
```

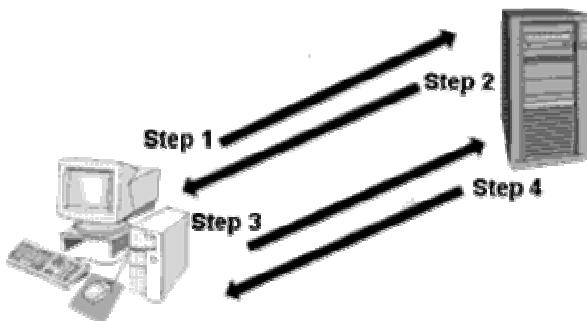
**List of some anonymous ftp sites:**

```
ftp.deakin.edu.au  
ftp.csn.net  
ftp.infomagic.nl
```

# Secure Socket Layer (SSL)

SSL (Secure Socket Layer) is a protocol for encrypting and decrypting data sent across direct internet connections. When a client makes an SSL connection with a server, all data sent to and from that server is encoded with a complex mathematical algorithm that makes it extremely difficult to decode anything that is intercepted.

The following is a step by step illustration of how SSL works.



**Step 1.** The client makes the initial connection with the server and requests that an SSL connection be made.

**Step 2.** If the server is properly configured, the server will send to the client its certificate and public key.

**Step 3.** The client compares the certificate from the server to a trusted authorities database. If the certificate is listed there, it means the client trusts the server and will move to step 4. If the certificate is not listed there, the user must add the certificate to the trusted authorities database before going to step 4.

**Step 4.** The client uses that public key to encrypt a session key and sends the session key to the server. If the server asks for the client's certificate in Step 2, the client must send it at this point.

**Step 5.** If the server is set up to receive certificates, it compares the certificate it received with those listed in its trusted authorities database and either accepts or rejects the connection.

If the connection is rejected, a fail message is sent to the client. If the connection is accepted, or if the server is not set up to receive certificates, it decodes the session key from the client with its own private key and sends a success message back to the client, thereby opening a secure data channel.

The key to understanding how SSL works is in understanding the parts that make SSL itself work. The following is a list of these parts and the role each plays.

**Client.** In this case, the client is WS\_FTP Pro.

**Certificate.** The Certificate file holds the identification information of the client or server. This file is used during connection negotiations to identify the parties involved. In some cases, the client's certificate must be signed by the server's certificate in order to open an SSL connection. Certificate files have the .crt ending.

**Session Key.** The session key is what both the client and the server use to encrypt data. It is created by the client.

**Public Key.** The public key is the device with which the client encrypts a session key. It does not exist as a file, but is a by-product of the creation of a certificate and private key. Data encrypted with a public key can only be decrypted by the private key that made it.

**Private Key.** The private key decrypts the client's session key that is encrypted by a public key. The private key file has the .key ending. Private keys should NEVER be distributed to anyone.

**Certificate Signing Request.** A certificate signing request is generated each time a certificate is created. This file is used when you need to have your certificate signed. Once the Certificate Signing Request file is signed, a new certificate is made and can be used to replace the unsigned certificate.

## How to make an SSL connection

To make an SSL connection with a server configured for SSL.

Be sure to select the Secure (SSL) option when you follow the directions for configuring a site.

After you click Connect, WS\_FTP Pro tells the server that you want to make an SSL connection. The server then transmits to you an identifying certificate, letting the client know who the server is. If that certificate is already listed in your Trusted Authority database, the connection is made.

If that certificate is not listed as a trusted authority, the Non-Trusted Authority dialog box appears.

Select the option you need and click **OK**. If the server does not require a certificate to be returned, the secure connection will be established. All data transmitted between you and the server will be encrypted.

If the server you are attempting to make a connection to asks WS\_FTP Pro to send back a certificate, follow the direction for Client Certificate Verification.

## Client Certificate Verification

If the server you are attempting to make a connection to requires your client to send an identifying certificate back to the server, you must:

1. Configure the site with the Secure (SSL) option selected.
2. Create a certificate. Refer to the section "Generating a Certificate" for more information.
3. Send the Certificate Signing Request file to your server administrator.
4. Once the server administrator signs the Certificate Signing Request, it will be sent back to you.
5. When you receive the file, follow the directions for "Selecting a Certificate", selecting the new certificate to go in the **Certificate** box.
6. Connect to the server.

## Generating a Certificate

To create an SSL certificate:

1. From the **Options** menu, select **Configure SSL**. The SSL Utilities window appears.
2. Click the **Certificate Creation** tab.



3. Enter a name in the **Certificate Set Name** box. This will be the name of the certificate that is generated by WS\_FTP Pro.
4. Click the **Browse (...)** button in the **Output Location** box to select the folder you want the certificate created in.
5. Enter information in all of the Certificate Information boxes:

**City/Town.** City or town where you are located. (Ex. Augusta)

**State/Province.** State or Province where you are located. (Ex. Georgia)

**Organization.** Company or individual user name.

**Common Name.** This can be either the name of the person creating the certificate or the fully qualified domain name of the server associated with the host.

**Pass Phrase.** Pass phrase that is to be used to encrypt the private key. It is important to remember this pass phrase. The pass phrase can be any combination of words, symbols, spaces, or numbers.

**Pass Phrase Confirmation.** Re-enter the same pass phrase as above.

**Country.** The country you are in. This must be a valid two letter country code. (Ex. US)

**E-mail.** E-mail address of the person the certificate belongs to.

**Unit.** Name of organizational unit. (Ex. Research and Development)

6. After all of the boxes are filled in correctly, click **Create** to generate the keys, certificate, and certificate signing request. If all of the boxes are not filled in, you cannot create the certificate.

If you are creating a certificate to be used by WS\_FTP Pro, you should send the certificate signing request (by E-mail) to your server administrator. If they require it, they will sign the certificate and return it to you. The returned certificate should be the one you identify in the Certificate Selection tab.

## Selecting a Certificate

The Certificate Selection tab is used to choose which private key and certificate you want to use during SSL connection negotiations. If a new certificate has not been created, follow the directions for "Generating a Certificate".



To select an SSL Certificate:

1. Click the **Browse (...)** button next to the **Private Key** box to select the private key you want to use during SSL negotiation.
2. Click the **Browse (...)** button next to the **Certificate** box to select the certificate you want to use during SSL negotiation. The certificate you use must have been created using the key you selected for the **Private Key** box.
3. Enter the pass phrase associated with that certificate in both the **Pass Phrase** and the **Pass Phrase Confirmation** boxes. A pass phrase can be any combination of words, symbols, spaces or numbers. It is case sensitive and must be written exactly the same way each time it is used.

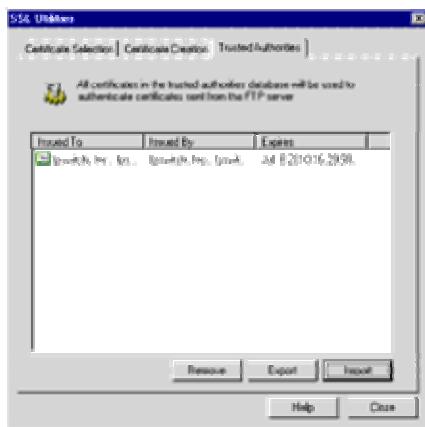
Without the correct pass phrase in both boxes, the certificate and private key cannot be verified and the selection cannot be saved.

4. Click **Apply** to save your entries.

Clicking the **Reset** button erases what you have done since the last time new settings were applied.

## Trusted Authorities

The Trusted Authorities tab stores a list of certificate names that are recognized by WS\_FTP Pro.



**Certificate Display:**

**Issued To.** Who the certificate was issued to.

**Issued By.** Who the certificate was signed by.

**Expires.** Date on which the certificate expires.

## Adding a Certificate

To add a certificate to the database:

1. Click the **Import** button and select the path and file name for the certificate. The **Add Certificate?** dialog box appears.



2. Review the information on that dialog box and click **Yes** to add the certificate to the database.

## Exporting a Certificate

To export a certificate from the Trusted Authorities database:

1. Select the certificate you want to copy out of your database.
2. Click the **Export** button.

Select the folder you want to copy the certificate to and enter the name you want to save the certificate file as.

3. Click **OK**.

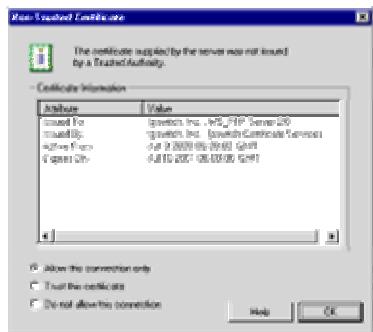
## Removing a Certificate

To remove a certificate:

1. Select the certificate to be removed.
2. Click **Remove**.
3. A warning appears advising you to export the certificate before you remove it. Removing the certificate deletes the certificate file.
4. Click **OK** to remove the certificate.

## Non-Trusted Certificate

When you connect to a server using the SSL connection option, that server sends you a certificate. If that certificate is not listed on the Trusted Authority tab, or if it was not signed by a certificate on this list, this dialog box appears.



#### Certificate Information:

**Issued To.** Name of the person or company who the certificate belongs to.

**Issued By.** Name of the person or company who signed the certificate.

**Active From.** The date on which this certificate was activated.

**Expires On.** The date the displayed certificate will no longer be a valid certificate.

#### Options

**Allow this connection only.** If this option is selected, the connection will be made, but WS\_FTP Pro will still not recognize the certificate as a trusted authority. The next time you attempt to connect to this server, this dialog box appears once again.

**Trust this certificate.** If this option is selected, the connection will be made and the certificate will be added to the trusted authority database in the Trusted Authority tab, so future connections can be made without you being prompted.

**Do not allow this connection.** If this option is selected, the connection will be terminated.

## SSH

SSH (Secure Shell) is a security protocol that allows you to make a secure connection to a server that has the SSH and SFTP (Secure File Transfer Protocol) protocols installed. Where FTP servers usually 'listen' on port 21 for connection, SSH servers use port 22.

Where SSL attempts to make a connection with unencrypted channels, SSH encrypts all communications to and from the client and server. When an SSH connection is made, SFTP is the protocol that is used to perform all tasks on that single secure connection.

## How to make an SSH connection

Making an SSH connection requires little additional configuration to new or your existing site profiles.

When creating a new site profile through the **New Site Wizard**, simply change the **Server Type** to **SFTP/SSH** when prompted by the wizard.

If editing an existing site profile:

1. Select the site from the **configured sites** list.
2. Click the **Edit** button.
3. Click the **Advanced** tab.
4. In the **Server type** pull-down, select **SFTP/SSH**.
5. Click **OK**.

When you use that profile to make a connection, the client will automatically attempt to make an SSH connection on port 22 of that server.

## Difference between SSH and SSL

With both SSL and SSH, it is up to the administrator of the server you are trying to make a connection with to tell you which server type is being run at that address. If you do not know, and attempt to make an SSL connection or an insecure connection to an SSH server, the connection will fail, even if you change the port to 22 in the Site Profile.

SSH (Secure Shell) and SSL (Secure Sockets Layer) can both be used to secure communications across the Internet. This page tries to explain the differences between the two in easily understood terms.

SSL was designed to secure web sessions; it can do more, but that's the original intent.

SSH was designed to replace telnet and FTP; it can do more, but that's the original intent.

SSL is a drop-in with a number of uses. It front-ends HTTP to give you HTTPS. It can also do this for POP3, SMTP, IMAP, and just about any other well-behaved TCP application. It's real easy for most programmers who are creating network applications from scratch to just grab an SSL implementation and bundle it with their app to provide encryption when communicating across the network via TCP.

SSH is a swiss-army-knife designed to do a lot of different things, most of which revolve around setting up a secure tunnel between hosts. Some implementations of SSH rely on

SSL libraries - this is because SSH and SSL use many of the same encryption algorithms (i.e. TripleDES).

SSH is not based on SSL in the sense that HTTPS is based on SSL. SSH does much more than SSL, and they don't talk to each other - the two are different protocols, but have some overlap in how they accomplish similar goals.

SSL by itself gives you nothing - just a handshake and encryption. You need an application to drive SSL to get real work done.

SSH by itself does a whole lot of useful stuff that allows users to perform real work. Two aspects of SSH are the console login (telnet replacement) and secure file transfers (ftp replacement), but you also get an ability to tunnel (secure) additional applications, enabling a user to run HTTP, FTP, POP3, and just about anything else THROUGH an SSH tunnel.

Without interesting traffic from an application, SSL does nothing. Without interesting traffic from an application, SSH brings up an encrypted tunnel between two hosts which allows you to get real work done through an interactive login shell, file transfers, etc

Last comment: HTTPS does not extend SSL, it uses SSL to do HTTP securely. SSH does much more than SSL, and you can tunnel HTTPS through it! Just because both SSL and SSH can do TripleDES doesn't mean one is based on the other.

# **TELNET**

Telnet stands for Telecommunications Network. Telnet is a program, which is preinstalled in windows 95/98/NT and UNIX operating systems. Telnet allows you to use your telephone line to gain access to and control of a remote **Internet PC/Server**.

When you telnet into a remote system it is as if you are sitting at a terminal and keyboard which is directly connected to the system. The commands, which you enter, are received as though they are from an individual who is sitting right at the system and typing the commands in directly and your monitor displays those exact results. You become in effect just another terminal on the host. You can be a thousand miles away from the system and perform functions on it as if you are sitting right there.

Telnet is a very useful protocol when properly used. Telnet has a number of advantages over FTP including the availability to chat with users, delete directories if it is not empty, and editing of files without downloading them from the server.

## **Telnet over TCP/IP**

On the Internet the ability to connect with another machine is made possible by the Transmission Control Protocol (TCP), which enables two machines to transmit data back and forth in a manner coherent to the operating systems of each device, and the Internet Protocol which provides a unique 32 bit address for each machine connected to the network. The telecommunications application built over these capabilities provides the local terminal with the means to emulate a terminal compatible with the remote computer.

### **Connection Establishment:**

The TELNET TCP connection is established between the user's port U and the server's port L. The server listens on its well known port L for such connections. Since a TCP connection is full duplex and identified by the pair of ports, the server can engage in many simultaneous connections involving its port L and different users ports U. Port assignment- When used for remote user access to service hosts, this protocol is assigned server port 23, that is L=23.

The TELNET protocol gives you the ability to connect to a machine, by giving commands and instructions interactively to that machine, thus creating an interactive connection. In such a case, the local system becomes transparent to the user, who gets the feeling that he is connected directly to the remote computer.

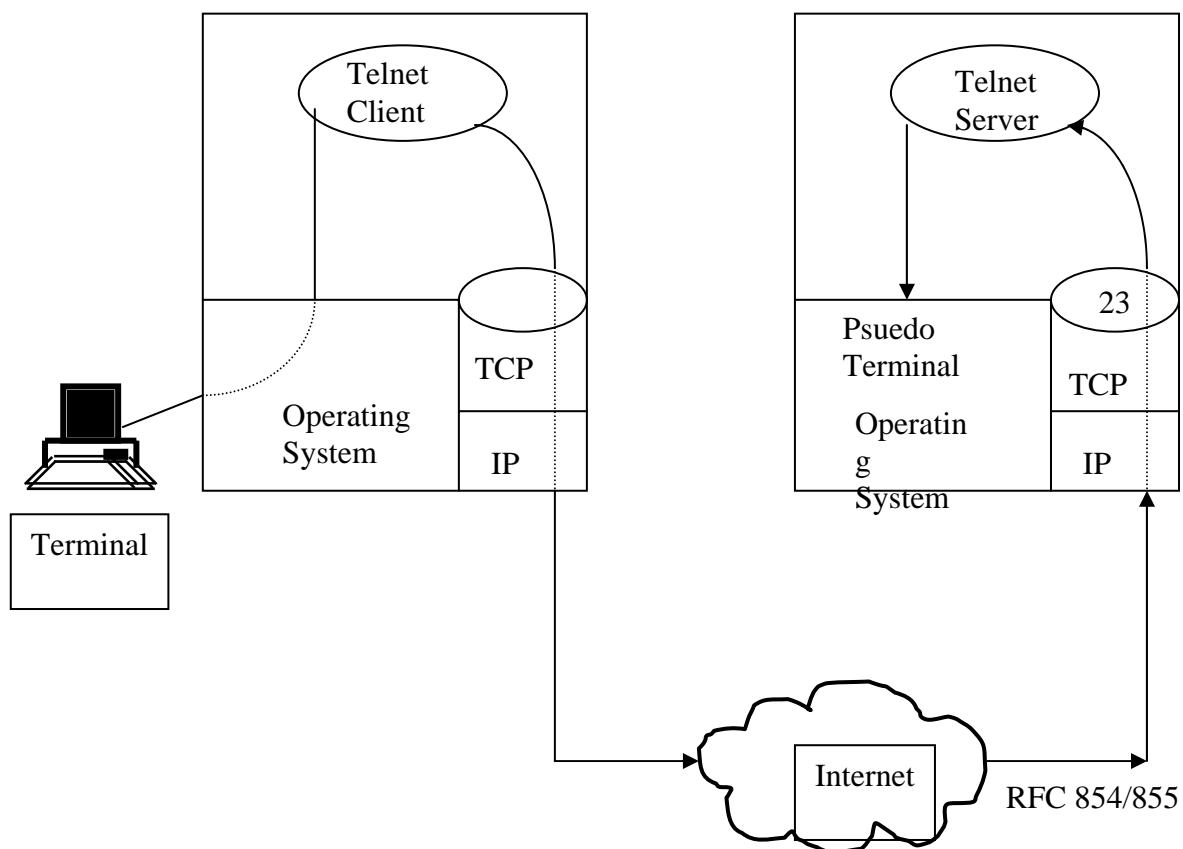
The commands typed by the user are transmitted directly to the remote machine and the response from the remote machine is displayed on the user's monitor screen. An interactive connection is also known as remote login.

In order to remote login the user's computer must have the ability to:

- Establish a connection to another machine.
- Emulate a terminal compatible with the remote machine.
- Regulate the flow of data from the user's terminal to remote machine

and vice-versa.

The figure below describes the path of data in a telnet remote terminal session as it travels from the user's keyboard to the remote operating system.



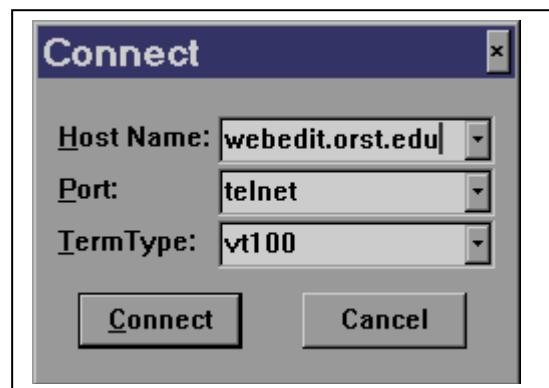


## How do I Telnet with a PC?

To Telnet it is necessary that you have the Windows telnet.exe program in your computer. First start the program by selecting 'Start', 'Run', and typing 'telnet'. Start a



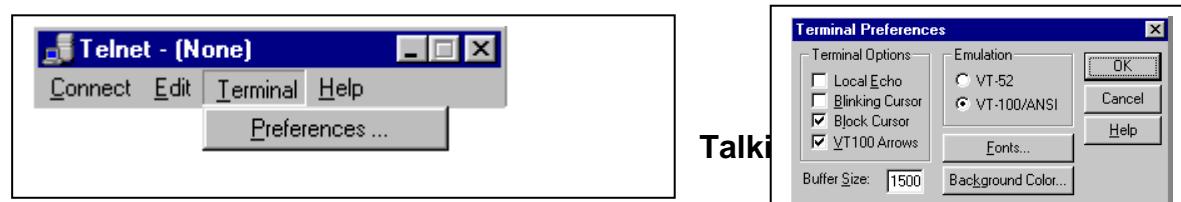
Telnet session by clicking on 'Connect' from the Menu bar and then select 'Remote System':



In the Host Name box enter the IP or Domain address of the remote computer. The other options should be set with 'Port: telnet', and the 'TermType: vt100'. This could be obtained by setting up the Terminal. Then click on Connect

## Setup Telnet

Before using telnet, set its buffer size. To do this click the menu Terminal and then Preferences and set the buffer size to 1500. Increasing the buffer size will allow more information to be stored in the buffer than the window pane allows. This way you can back scroll through the information that naturally scrolls off the regular window pane.

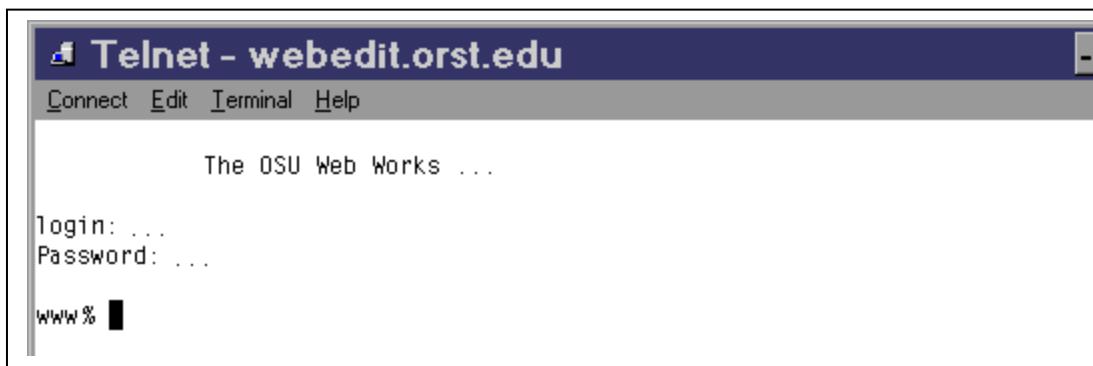


Talki

To talk to any user on

the system, just type "talk login" where login is the actual login of the user you want to talk to. If the user accepts your chat request, a divided screen will appear and you two will be able to chat. Once you have telneted onto a server you are able to invoke whatever UNIX or PERL commands are accepted by that system..

Once you connect to the remote computer you will enter your user name followed by your password.



When you have successfully logged in you will see a command prompt. The command prompt is where you send your commands to the remote computer.

### **Expired Passwords:**

If your password has expired you should get the following messages:

Choose a new password.

Enter login password:

This means you MUST enter your OLD password a second time. Then you will be prompted to enter a new password two times.

### **Basic Telnet commands**

Telnet comes with a large number of commands. Below we attempt to address basic telnet commands.

ls

This command will list all of the files and directories within the current directory.

cd <directory>

This command will change your current directory to the directory you specify. An example would be that "cd public" would take you into your public directory.

mv <filename>

This command will move a file from its current location within the current directory to the directory you specify. For example, lets say you are in your public directory and you want to move the file "links.cgi" into your "cgi-local" directory. At the prompt type "mv links.cgi cgi-local".

chmod XXX

This command will set the permissions on a file or directory to whatever you specify. The "XXX" would be

<filename or  
directory>

replaced by actual numbers, such as 644. For example, if you wanted to set the permissions on links.cgi file in your cgi-local directory, you would first go into that directory then type "chmod 755 links.cgi" at the prompt.

rm <filename or  
directory>

This command will delete the filename or directory you specify in the current directory. If you want to remove the directory named "user", you type "rm user" at the prompt.

man <command>

This is one of the most helpful commands for new users. It allows you to see all of the different options for a particular command. For example, if you type "man ls", you will see all of the different options available for the list (ls) command.

Traceroute  
<domain>

This command will perform a traceroute on a particular virtual domain to see how many hops it takes to get from your location to the domain specified. An example would be "traceroute yahoo.com", which would show how long it takes for packets to get to yahoo.com and how they get there.

## Troubleshooting:

Nothing happens when you try to connect to a telnet site: The site could be down for maintenance or problems. You get a "host unavailable" message: The Telnet site is down for some reason. Try again later. You get a "host unknown" message: Check your spelling of the site name. You type in a password on a Telnet site that requires one, and you get a "login incorrect" message: Try logging in again. If you get the message again hit your control and '[' keys i.e. (^]) at the same time to disengage and return to your host system.

- You can't seem to disconnect from the Telnet site: Use ^] keys to disengage and return to your host system.
- You end up with a screen full of garbage: Chances are you have chosen an incorrect terminal emulation setting. Disconnect and try again.

## My SQL

The database has become an integral part of almost every human's life. Without it, many things we do would become very tedious, perhaps impossible tasks. Banks, universities, and libraries are three examples of organizations that depend heavily on some sort of database system. On the Internet, search engines, online shopping, and even the website naming convention (<http://www...>) would be impossible without the use of a database. A database that is implemented and interfaced on a computer is often termed a database server.

One of the fastest SQL (Structured Query Language) database servers currently on the market is the MySQL server, developed by T.c.X. DataKonsultAB. MySQL, available for download at <http://www.mysql.com>, offers the database programmer with an array of options and capabilities rarely seen in other database servers. What's more, MySQL is free of charge for those wishing to use it for private and commercial use. Those wishing to develop applications specifically using MySQL should consult MySQL's licensing section, as there is a charge for licensing the product.

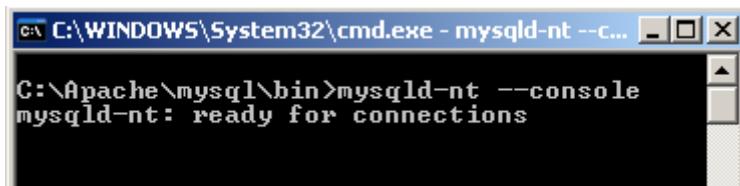
These capabilities range across a number of topics, including the following:

- Ability to handle an unlimited number of simultaneous users.
- Capacity to handle 50,000,000+ records.
- Very fast command execution, perhaps the fastest to be found on the market.
- Easy and efficient user privilege system.

However, perhaps the most interesting characteristic of all is the fact that it's **free**. That's right, T.c.X offers MySQL as a free product to the general public.

### Starting MySQL

Before starting the MySQL client make sure the server is turned on. If you run Windows 9x start the mysqld.exe or if you run Windows 2000/XP run the mysqld-nt.exe



The --console option tells mysqld-nt not to remove the console window. If you are running Windows NT/2000/XP it's better to install MySQL as a service. That way mysql server will be automatically started when you start Windows.

Use mysqld-nt --install to install mysqld as a service and mysqld-nt --remove to remove mysqld from the service list

Once the server is on, open another DOS window and type mysql, you should see something like this

```
C:\>mysql  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 3 to server version: 4.0.18-nt  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
  
mysql>
```

Or if you need to specify a user name and password you can start mysql like this :

```
C:\>mysql -h localhost -u root -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 4 to server version: 4.0.18-nt  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
  
mysql>
```

You can also specify the database you want to use. If you already have a database named petstore you can start mysql as

```
C:\>mysql petstore  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 5 to server version: 4.0.18-nt  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
  
mysql>
```

Once you're done you can end mysql client by typing **quit** or **exit** at the mysql> prompt

```
mysql> exit  
Bye  
  
C:\>
```

**Note :** If you get this kind of error message when trying to run mysql from the DOS window

```
C:\>mysql  
'mysql' is not recognized as an internal or external command,  
operable program or batch file.
```

that means you haven't set the path to mysql bin directory. To solve the problem you can add the following line to your autoexec.bat file :

```
path=%path%;c:\mysql\bin
```

assuming that you install MySQL in c:\mysql. Or if you're using Windows XP you can go to :

Start->Settings->Control Panel->System->Advanced->Environment Variables

Chose Edit on the System Variables section and add c:\mysql\bin to the path environment variable.

## Connect to MySQL Database

Opening a connection to MySQL database from PHP is easy. Just use the mysql\_connect() function like this

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = 'password';

$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die
    ('Error connecting to mysql');

$dbname = 'petstore';
mysql_select_db($dbname);
?>
```

**\$dbhost** is the name of MySQL server. When your webserver is on the same machine with the MySQL server you can use localhost or 127.0.0.1 as the value of **\$dbhost**. The **\$dbuser** and **\$dbpass** are valid MySQL user name and password.

For adding a user to MySQL visit this page : [MySQL Tutorial](#)

Don't forget to select a database using **mysql\_select\_db()** after connecting to mysql. If no database selected your query to select or update a table will not work.

Sometimes a web host will require you to specify the MySQL server name and port number. For example if the MySQL server name is **db.php-mysql-tutorial.com** and the port number **is 3306** (the default port number for MySQL) then you can modify the above code to :

```
<?php
$dbhost = 'db.php-mysql-tutorial.com:3306';
$dbuser = 'root';
$dbpass = 'password';

$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die
    ('Error connecting to mysql');

$dbname = 'petstore';
mysql_select_db($dbname);
?>
```

It's a common practice to place the routine of opening a database connection in a separate file. Then everytime you want to open a connection just **include** the file. Usually the host, user, password and database name are also separated in a configuration file.

An example of **config.php** that stores the connection configuration and **opendb.php** that opens the connection are :

Source code : **config.php**, **opendb.php**

```
<?php
// This is an example of config.php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = 'password';
$dbname = 'phpcake';
?>
```

```
<?php
// This is an example opendb.php
$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die
    ('Error connecting to mysql');
mysql_select_db($dbname);
?>
```

So now you can open a connection to mysql like this :

```
<?php
include 'config.php';
include 'opendb.php';

// ... do something like insert or select, etc

?>
```

## Create MySQL Database With PHP

### *Closing the Connection*

The connection opened in a script will be closed as soon as the execution of the script ends. But it's better if you close it explicitly by calling **mysql\_close()** function. You could also put this function call in a file named **closedb.php**.

Source code : **closedb.php**

```
<?php
// an example of closedb.php
// it does nothing but closing
// a mysql database connection

mysql_close($conn);
?>
```

Now that you have put the database configuration, opening and closing routines in separate files your PHP script that uses mysql would look something like this :

```
<?php  
include 'config.php';  
include 'opendb.php';  
  
// ... do something like insert or select, etc  
  
include 'closedb.php';  
?>
```

To create a database use the **mysql\_query()** function to execute an SQL query like this

```
<?php  
include 'config.php';  
include 'opendb.php';  
  
$query = "CREATE DATABASE phpcake";  
$result = mysql_query($query);  
  
include 'closedb.php';  
?>
```

Please note that the query **should not** end with a semicolon.

PHP also provide a function to create MySQL database, **mysql\_create\_db()**. This function is deprecated though. It is better to use **mysql\_query()** to execute an SQL CREATE DATABASE statement instead like the above example.

If you want to create MySQL database using PHP **mysql\_create\_db()** function you can do it like this :

```
<?php  
include 'config.php';  
include 'opendb.php';  
  
mysql_create_db('phpcake');  
  
include 'closedb.php';  
?>
```

If you want to create tables in the database you just created don't forget to call **mysql\_select\_db()** to access the new database.

**Note:** some webhosts require you to create a MySQL database and user through your website control panel (such as CPanel). If you get an error when trying to create database this might be the case.

## ***Creating the Tables***

To create tables in the new database you need to do the same thing as creating the database. First create the SQL query to create the tables then execute the query using **mysql\_query()** function.

Example : contact.php

Source code : **contact.php**

```
<?php
include 'config.php';
include 'opendb.php';

$query = 'CREATE DATABASE phpcake';
$result = mysql_query($query);

mysql_select_db('phpcake') or die('Cannot select database');

$query = 'CREATE TABLE contact(
    'cid INT NOT NULL AUTO_INCREMENT,
    'cname VARCHAR(20) NOT NULL,
    'cemail VARCHAR(50) NOT NULL,
    'csubject VARCHAR(30) NOT NULL,
    'cmessage TEXT NOT NULL,
    'PRIMARY KEY(cid))';

$result = mysql_query($query);

include 'closedb.php';
?>
```

Of course when you need to create lots of tables it's a good idea to read the query from a file then save in **\$query** variable instead of writing the query in your script.

```
<?php
include 'config.php';
include 'opendb.php';

$queryFile = 'myquery.txt';

$fp = fopen($queryFile, 'r');
$query = fread($fp, filesize($queryFile));
fclose($fp);
$result = mysql_query($query);

include 'closedb.php';
?>
```

## ***Deleting a Database***

As with creating a database, it is also preferable to use **mysql\_query()** and to execute the SQL DROP DATABASE statement instead of using **mysql\_drop\_db()**

```

<?php
include 'config.php';
include 'opendb.php';

// ... do something here

$query = 'DROP DATABASE phpcake';
$result = mysql_query($query);

// ... probably do something here too

include 'closedb.php';
?>

```

After the database and the tables are ready it's time to put something into the database.

## Insert Data To MySQL Database

Inserting data to MySQL is done by using **mysql\_query()** to execute INSERT query. Note that the query string **should not** end with a semicolon. Below is an example of adding a new MySQL user by inserting a new row into table `user` in database `mysql` :

Example : insert.php

Source code : [insert.php](#)

```

<?php
include 'library/config.php';
include 'library/opendb.php';

mysql_select_db($mysql);
$query = "INSERT INTO user (host, user, password, select_priv, insert_priv, update_
priv) VALUES ('localhost', 'phpcake', PASSWORD('mypass'), 'Y', 'Y', 'Y')";

mysql_query($query) or die('Error, insert query failed');

$query = "FLUSH PRIVILEGES";
mysql_query($query) or die('Error, insert query failed');

include 'library/closedb.php';
?>

```

In the above example **mysql\_query()** was followed by **die()**. If the query fail the error message will be printed and the script's execution is terminated. Actually you can use **die()** with any function that might not execute properly. That way you can be sure that the script won't continue to run when an error occurred.

In a real application the values of an `INSERT` statement will be form values. As a safe precaution always escape the values using **addslashes()** if **get\_magic\_quotes\_gpc()** returns false. Below is an example of using form values with `INSERT`. It's the same as above except that the new username and password are taken from `$_POST` :

Example : adduser.php  
Source code : **adduser.php**

```
<html>
<head>
<title>Add New MySQL User</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
<?php
if(isset($_POST['add']))
{
include 'library/config.php';
include 'library/opendb.php';

$username = $_POST['username'];
$password = $_POST['password'];

$query = "INSERT INTO user (host, user, password, select_priv, insert_priv,
update_priv) VALUES ('localhost', '$username', PASSWORD('"$password'), 'Y', 'Y',
'Y')";
mysql_query($query) or die('Error, insert query failed');

$query = "FLUSH PRIVILEGES";
mysql_query($query) or die('Error, insert query failed');

include 'library/closedb.php';
echo "New MySQL user added";
}
else
{
?>
<form method="post">
<table width="400" border="0" cellspacing="1" cellpadding="2">
<tr>
<td width="100">Username</td>
<td><input name="username" type="text" id="username"></td>
</tr>
<tr>
<td width="100">Password</td>
<td><input name="password" type="text" id="password"></td>
</tr>
<tr>
<td width="100">&ampnbsp</td>
<td>&ampnbsp</td>
</tr>
<tr>
<td width="100">&ampnbsp</td>
<td><input name="add" type="submit" id="add" value="Add New User"></td>
</tr>
</table>
</form>
<?php
}
?>
```

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

## Get Data From MySQL Database

Using PHP you can run a MySQL SELECT query to fetch the data out of the database. You have several options in fetching information from MySQL. PHP provide several functions for this. The first one is **mysql\_fetch\_array()** which fetch a result row as an associative array, a numeric array, or both.

Below is an example of fetching data from MySQL, the table contact have three columns, name, subject and message.

Example : select.php

Source code : **select.php**, **contact.txt**

```
<?php
include 'config.php';
include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_array($result, MYSQL_ASSOC))
{
    echo "Name : {$row['name']} <br> .
        "Subject : {$row['subject']} <br> .
        "Message : {$row['message']} <br><br>";
}

include 'closedb.php';
?>
```

The while() loop will keep fetching new rows until mysql\_fetch\_array() returns FALSE, which means there are no more rows to fetch. The content of the rows are assigned to the variable \$row and the values in row are then printed. Always remember to put curly brackets when you want to insert an array value directly into a string.

In above example I use the constant **MYSQL\_ASSOC** as the second argument to **mysql\_fetch\_array()**, so that it returns the row as an associative array. With an associative array you can access the field by using their name instead of using the index . Personally I think it's more informative to use **\$row['subject']** instead of **\$row[1]**.

PHP also provide a function called **mysql\_fetch\_assoc()** which also return the row as an associative array.

```
<?php
include 'config.php';
include 'opendb.php';
```

```

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_assoc($result))
{
    echo "Name : {$row['name']} <br> .
        "Subject : {$row['subject']} <br> .
        "Message : {$row['message']} <br><br>";
}

include 'closedb.php';
?>

```

You can also use the constant **MYSQL\_NUM**, as the second argument to **mysql\_fetch\_array()**. This will cause the function to return an array with numeric index.

```

<?php
include 'config.php';
include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_array($result, MYSQL_NUM))
{
    echo "Name :{$row[0]} <br> .
        "Subject : {$row[0]} <br> .
        "Message : {$row[0]} <br><br>";
}

include 'closedb.php';
?>

```

Using the constant **MYSQL\_NUM** with **mysql\_fetch\_array()** gives the same result as the function **mysql\_fetch\_row()**.

There is another method for you to get the values from a row. You can use **list()**, to assign a list of variables in one operation.

```

<?php
include 'config.php';
include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while(list($name,$subject,$message)= mysql_fetch_row($result))
{
    echo "Name :$name <br> .
        "Subject : $subject <br> .
        "Message : $row <br><br>";
}

```

```
}
```

```
include 'closedb.php';
?>
```

In above example, **list()** assign the values in the array returned by **mysql\_fetch\_row()** into the variable **\$name**, **\$subject** and **\$message**.

Of course you can also do it like this

```
<?php
include 'config.php';
include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_row($result))
{
    $name = $row[0];
    $subject = $row[1];
    $message = $row[2];
    echo "Name : $name <br>" .
        "Subject : $subject <br>" .
        "Message : $row <br><br>";
}

include 'closedb.php';
?>
```

So you see you have lots of choices in fetching information from a database. Just choose the one appropriate for your program

### Freeing the memory ?

In some cases a query can return large result sets. As these results are stored in memory there's a concern about memory usage. However you do not need to worry that you will have to call this function in all your script to prevent memory congestion. In PHP all results memory is automatically freed at the end of the script's execution.

But you are really concerned about how much memory is being used for queries that return large result sets you can use **mysql\_free\_result()**. Calling this function will free all memory associated with the result identifier (**\$result**).

Using the above example you can call **mysql\_free\_result()** like this :

```
<?php
include 'config.php';
include 'opendb.php';
```

```

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_row($result))
{
...
}

mysql_free_result($result);

include 'closedb.php';

?>

```

### Convert MySQL Query Result To Excel

Using PHP to convert MySQL query result to Excel format is also common especially in web based finance applications. The finance data stored in database are downloaded as Excel file for easy viewing. There is no special functions in PHP to do the job. But you can do it easily by formatting the query result as tab separated values or put the value in an HTML table. After that set the content type to application/vnd.ms-excel

Example : **convert.php**

Source : **convert.php, students.txt**

```

<?php
include 'library/config.php';
include 'library/opendb.php';

$query = "SELECT fname, lname FROM students";
$result = mysql_query($query) or die('Error, query failed');

$tsv = array();
$html = array();
while($row = mysql_fetch_array($result, MYSQL_NUM))
{
    $tsv[] = implode("\t", $row);
    $html[] = "<tr><td>" . implode("</td><td>", $row) . "</td></tr>";
}

$tsv = implode("\r\n", $tsv);
$html = "<table>" . implode("\r\n", $html) . "</table>";

$fileName = 'mysql-to-excel.xls';
header("Content-type: application/vnd.ms-excel");
header("Content-Disposition: attachment; filename=$fileName");

```

```

echo $tsv;
//echo $html;

include 'library/closedb.php';
?>

```

In the above example `$tsv` is a string containing tab separated values and `$html` contain an HTML table. I use `implode()` to join the values **of \$row** with tab to create a tab separated string.

After the while loop `implode()` is used once again to join the rows using newline characters. The headers are set and the value of `$tsv` is then printed. This will force the browser to save the file as `mysql-to-excel.xls`

Try running the script in your own computer then try commenting `echo $tsv` and uncomment `echo $html` to see the difference.

## Using Paging

Ever use a Search Engine? I'm sure you have, lots of time. When Search Engines found thousands of results for a keyword do they spit out all the result in one page? Nope, they use paging to show the result little by little.

Paging means showing your query result in **multiple pages** instead of just put them all in one long page. Imagine waiting for five minutes just to load a search page that shows 1000 result. By splitting the result in multiple pages you can save download time plus you don't have much scrolling to do.

To show the result of a query in several pages first you need to know **how many rows you have** and **how many rows per page** you want to show. For example if I have 295 rows and I show 30 rows per page that mean I'll have ten pages (rounded up).

For the example I created a table named `randoms` that store 295 random numbers. Each page shows 20 numbers.

Example: `paging.php`

Source code :`paging.php`

```

<?php
include 'library/config.php';
include 'library/opendb.php';

// how many rows to show per page
$rowsPerPage = 20;

// by default we show first page
$pageNum = 1;

// if $_GET['page'] defined, use it as page number
if(isset($_GET['page']))
{

```

```

    $pageNum = $_GET['page'];
}

// counting the offset
$offset = ($pageNum - 1) * $rowsPerPage;

$query = " SELECT val FROM randoms " .
    " LIMIT $offset, $rowsPerPage";
$result = mysql_query($query) or die('Error, query failed');

// print the random numbers
while($row = mysql_fetch_array($result))
{
    echo $row['val'] . '<br>';
}

// ... more code here
?>

```

Paging is implemented in MySQL using LIMIT that take two arguments. The first argument specifies the offset of the first row to return, the second specifies the maximum number of rows to return. The offset of the first row is 0 ( not 1 ).

When **paging.php** is called for the first time the value of **`$_GET['page']`** is not set. This caused \$pageNum value to remain 1 and the query is :

```
SELECT val FROM randoms LIMIT 0, 20
```

which returns the first 20 values from the table. But when paging.php is called like this <http://www.php-mysql-tutorial.com/examples/paging/paging.php?page=4> the value of \$pageNum becomes 4 and the query will be :

```
SELECT val FROM randoms LIMIT 60, 20
```

this query returns rows 60 to 79.

After showing the values we need to print the links to show any pages we like. But first we have to count the number of pages. This is achieved by dividing the number of total rows by the number of rows to show per page :

```
$maxPage = ceil($numrows/$rowsPerPage);
```

```
<?php
// ... the previous code

// how many rows we have in database
$query = "SELECT COUNT(val) AS numrows FROM randoms";
$result = mysql_query($query) or die('Error, query failed');
$row = mysql_fetch_array($result, MYSQL_ASSOC);
$numrows = $row['numrows'];

// how many pages we have when using paging?
$maxPage = ceil($numrows/$rowsPerPage);
```

```

// print the link to access each page
$self = $_SERVER['PHP_SELF'];
$nav = "";

for($page = 1; $page <= $maxPage; $page++)
{
if ($page == $pageNum)
{
    $nav .= " $page "; // no need to create a link to current page
}
else
{
    $nav .= " <a href=\"$self?page=$page\">$page</a> ";
}
}

// ... still more code coming
?>

```

The mathematical function **ceil()** is used to round up the value of **\$numrows/\$rowsPerPage**.

In this case the value of total **rows \$numrows** is 295 and **\$rowsPerPage** is 20 so the result of the division is 14.75 and by using **ceil()** we get **\$maxPage = 15**

Now that we know how many pages we have we make a loop to print the link. Each link will look something like this:

```
<a href="paging.php?page=5">5</a>
```

You see that we use **\$\_SERVER['PHP\_SELF']** instead of **paging.php** when creating the link to point to the paging file. This is done to avoid the trouble of modifying the code in case we want to change the filename.

We are almost complete. Just need to add a little code to create a 'Previous' and 'Next' link. With these links we can navigate to the previous and next page easily. And while we at it let's also create a 'First page' and 'Last page' link so we can jump straight to the first and last page when we want to.

```

<?php
// ... the previous code

// creating previous and next link
// plus the link to go straight to
// the first and last page

if ($pageNum > 1)
{
    $page = $pageNum - 1;
    $prev = " <a href=\"$self?page=$page\">[Prev]</a> ";

    $first = " <a href=\"$self?page=1\">[First Page]</a> ";
}

```

```

}

else
{
    $prev = ' '; // we're on page one, don't print previous link
    $first = ' '; // nor the first page link
}

if ($pageNum < $maxPage)
{
    $page = $pageNum + 1;
    $next = " <a href=\"$self?page=$page\">[Next]</a> ";

    $last = " <a href=\"$self?page=$maxPage\">[Last Page]</a> ";
}
else
{
    $next = ' '; // we're on the last page, don't print next link
    $last = ' '; // nor the last page link
}

// print the navigation link
echo $first . $prev . $nav . $next . $last;

// and close the database connection
include '../library/closedb.php';

// ... and we're done!
?>

```

Making these navigation link is actually easier than you may think. When we're on the fifth page we just make the 'Previous' link point to the fourth. The same principle also apply for the 'Next' link, we just need to add one to the page number.

One thing to remember is that we don't need to print the 'Previous' and 'First Page' link when we're already on the first page. Same thing for the 'Next' and 'Last' link. If we do print them that would only confuse the one who click on it. Because we'll be giving them the exact same page.

### We got a problem here...

Take a look at **this slightly modified version** of `paging.php`. Instead of showing 20 numbers in a page, I decided to show just three.

See the problem already?

Those page numbers are running across the screen! Yuck!

This call for a little modification to the code. Instead of printing the link to each and every page we will just saying something like "Viewing page 4 of 99 pages".

Than means we have to remove these code :

```

<?php
// ... the previous code

$nav = "";

for($page = 1; $page <= $maxPage; $page++)
{
    if ($page == $pageNum)
    {
        $nav .= " $page "; // no need to create a link to current page
    }
    else
    {
        $nav .= " <a href=\"$self?page=$page\">$page</a> ";
    }
}

// ... the rest here
?>

```

And then modify this one

```

<?php
// ...

// print the navigation link
echo $first . $prev . $nav . $next . $last;

// ...
?>

```

Into this

```

<?php
// ...

// print the navigation link
echo $first . $prev .
" Showing page $pageNum of $maxPage pages " . $next . $last;

// ...
?>

```

Take a look at the result here, and you can get the source code [here](#).

## Using Paging ( Part 2 )

When there's more than one column involved in paging there isn't much that we need to modify. We only need to decide how to count the total number of rows we have in the table. Consider the student table. This table have five columns as shown in the SQL below.

Source : **examples/source/student.sql**

```
CREATE TABLE student(
    id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(30) NOT NULL,
    address VARCHAR(50) NOT NULL,
    age TINYINT UNSIGNED NOT NULL,
    register_date DATE NOT NULL,
    PRIMARY KEY (id)
);
```

In the select query we just select all the columns. You can also use SELECT \* instead of mentioning all the column names ( SELECT id, name, address, age, register\_date ). But personally i prefer writing the column names in the query so that by reading the code i know what the column names in a table without having to check the database.

Example: **paging4.php**

Source code :**paging4.php**

```
<?php
include 'library/config.php';
include 'library/opendb.php';

// how many rows to show per page
$rowsPerPage = 3;

// by default we show first page
$pageNum = 1;

// if $_GET['page'] defined, use it as page number
if(isset($_GET['page']))
{
    $pageNum = $_GET['page'];
}

// counting the offset
$offset = ($pageNum - 1) * $rowsPerPage;

$query = "SELECT id, name, address, age, register_date
          FROM student
          LIMIT $offset, $rowsPerPage";
$result = mysql_query($query) or die('Error, query failed');

// print the student info in table
echo '<table border="1"><tr><td>Student
Id</td><td>Name</td><td>Address</td><td>Age</td><td>Register
Date</td></tr>';
while(list($id, $name, $address, $age, $regdate) = mysql_fetch_array($result))
{
    echo "<tr><td>$id</td><td>$name</td><td>$address</td>
```

```

<td>$age</td><td>$regdate</td></tr>";
}
echo '</table>';
echo '<br>';

// ... more code here
?>

```

In this example we print the result in table. Before looping through the array we just echo the starting table code and the header which displays the column names. Then in the loop we just print the values in a HTML table row.

The next thing is finding out the total number of rows. There are several ways to do it. The first one is shown below. It's the same method used in previous examples. We just use the COUNT() function

Example: **paging4.php**

Source code :**paging4.php**

```

<?php
// ... previous code here

// how many rows we have in database
$query = "SELECT COUNT(id) AS numrows FROM student";
$result = mysql_query($query) or die('Error, query failed');
$row = mysql_fetch_array($result, MYSQL_ASSOC);
$numrows = $row['numrows'];

// ... just the same code that prints the prev & next link
?>

```

You can also count any other columns since they all yield the same result. So your query can be rewritten into this :

```

<?php
// ...
$query = "SELECT COUNT(name) AS numrows FROM student";
// ...
?>

```

Or this :

```

<?php
// ...
$query = "SELECT COUNT(age) AS numrows FROM student";
// ...
?>

```

There is another way to count the total rows. Instead of using COUNT() function in the query you use a simple SELECT <column> and use **mysql\_num\_rows()** to see how many rows returned.

Take a look at the code below. We now separate the query into two parts. One is the normal SELECT query and the second is the SQL that performs the paging. After finish printing the result you can reuse the first part of the query to find the total number of rows.

Example: **paging5.php**  
Source code :**paging5.php**

```
<?php
// ... same old code to get the page number and counting the offset

$query = "SELECT id, name, address, age, register_date
          FROM student ";

$pagingQuery = "LIMIT $offset, $rowsPerPage";
$result = mysql_query($query . $pagingQuery) or die('Error, query failed');

// ... the code that prints the result in a table

// how many rows we have in database
$result = mysql_query($query) or die('Error, query failed');
$numrows = mysql_num_rows($result);

// ... and here is the code that print the prev & next links
?>
```

There is another advantage in separating the original query with the paging query. In case you only wish to list all student whose age is older than 15. You just need to modify the original query and you don't have to worry about changing the query to find the total number of rows. The example is shown below :

```
<?php
// ... same old code to get the page number and counting the offset

$query = "SELECT id, name, address, age, register_date
          FROM student
          WHERE age > 15";

$pagingQuery = "LIMIT $offset, $rowsPerPage";
$result = mysql_query($query . $pagingQuery) or die('Error, query failed');

// ... the code that prints the result in a table

// how many rows we have in database
$result = mysql_query($query) or die('Error, query failed');
$numrows = mysql_num_rows($result);

// ... and here is the code that print the prev & next links
?>
```

The disadvantage of this method is that the second execution of **mysql\_query()** will retrieve all columns from the database. This is very useless since we're not going to use them. We only interested in finding the

total rows returned by that query. In the end it's really up to you to decide which method you prefer.

## MySQL Update and Delete

There are no special ways in PHP to perform update and delete on MySQL database. You still use **mysql\_query()** to execute the UPDATE or DELETE statement.

For instance to update a password in mysql table for username phpcake can be done by executing an **UPDATE** statement with **mysql\_query()** like this:

Example : update.php

Source code : **update.php**

```
<?php
include 'library/config.php';
include 'library/opendb.php';

mysql_select_db('mysql')
or die('Error, cannot select mysql database');

$query = "UPDATE user SET password = PASSWORD('newpass')". "WHERE user =
'phpcake'";

mysql_query($query) or die('Error, query failed');
include 'library/closedb.php';
?>
```

There is one important thing that you should be aware of when updating and deleting rows from database. That is **data integrity**.

If you're using **InnoDB** tables you can leave the work of maintaining data integrity to MySQL . However when you're using other kind of tables you need to enforce the data integrity manually.

To make sure that your update and delete queries will not break the data integrity. You have to make appropriate update and delete queries for all tables referencing to the table you update or delete.

For example, suppose you have two tables, Class and Student. The Student table have a foreign key column, cid which references to the class\_id column in table Class. When you want to update a class\_id in Class table you will also need to update the cid column in Student table to maintain data integrity.

Suppose i want to change the class\_id of Karate from 3 to 10. Since there is a row in Student table with cid value of 3, I have to update that row too.

```
$query = "UPDATE Class SET class_id = 10 WHERE class_id = 3";
mysql_query($query);
```

```
$query = "UPDATE Student SET cid = 10 WHERE cid = 3";
mysql_query($query);
```

Below are the data in Table and Student class before an update query :

<b>Table Class</b>	
<b>class_id</b>	<b>class_name</b>
1	Silat
2	Kungfu
3	Karate
4	Taekwondo

<b>Table Student</b>		
<b>student_id</b>	<b>student_name</b>	<b>cid</b>
1	Uzumaki Naruto	1
2	Uchiha Sasuke	3
3	Haruno Sakura	2

Now the content of Table and Student class after the update query are:

<b>Table Class</b>	
1	Silat

<b>class_id</b>	<b>class_name</b>
1	Silat
2	Kungfu
10	Karate
4	Taekwondo

**Table Student**

<b>student_id</b>	<b>student_name</b>	<b>cid</b>
1	Uzumaki Naruto	1
2	Uchiha Sasuke	10
3	Haruno Sakura	2

You can go as far as creating your own functions in PHP to ensure the data integrity. I have done this before and I hope you don't do it. Save yourself the headache and just write appropriate queries to maintain your data integrity whenever you update / delete rows from a table.

This means that whenever you make a query to update / delete always consult your database design to see if you need to update / delete another table to maintain data integrity. Your code will be more portable like this.

