WEB APPLICATION DEVELOPMENT LANGUAGES

HTML

What is HTML?

HTML is a language for describing web pages.

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is not a programming language, it is a **markup language**
- A markup language is a set of **markup tags**
- HTML uses **markup tags** to describe web pages

HTML Tags

HTML markup tags are usually called HTML tags

- HTML tags are keywords surrounded by **angle brackets** like <html>
- HTML tags normally **come in pairs** like and
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- Start and end tags are also called **opening tags** and **closing tags**

HTML Documents = Web Pages

- HTML documents **describe web pages**
- HTML documents **contain HTML tags** and plain text
- HTML documents are also **called web pages**

The purpose of a web browser (like Internet Explorer or Firefox) is to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page:

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

- The text between <html> and </html> describes the web page
- The text between <body> and </body> is the visible page content
- The text between <h1> and </h1> is displayed as a heading
- The text between and is displayed as a paragraph

Editing HTML

HTML can be written and edited using many different editors like Dreamweaver and Visual Studio.

However, in this tutorial we use a plain text editor (like Notepad) to edit HTML. We believe using a plain text editor is the best way to learn HTML.

.HTM or .HTML File Extension?

When you save an HTML file, you can use either the .htm or the .html file extension. There is no difference, it is entirely up to you.

HTML Headings
HTML headings are defined with the <h1> to <h6> tags.
Example
```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

HTML Paragraphs
HTML paragraphs are defined with the tag.
Example
```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

HTML Links
HTML links are defined with the <a> tag.
Example
```
<a href="http://www.w3schools.com">This is a link</a>
```

HTML Images
HTML images are defined with the <img> tag.
Example
```
<img src="w3schools.jpg" width="104" height="142" />
```

HTML Elements
An HTML element is everything from the start tag to the end tag:

Start tag *	Element content	End tag *
	This is a paragraph	
	This is a link	

***** The start tag is often called the **opening tag**. The end tag is often called the **closing tag**.

HTML Element Syntax

- An HTML element starts with a **start tag / opening tag**
- An HTML element ends with an **end tag / closing tag**
- The **element content** is everything between the start and the end tag
- Some HTML elements have **empty content**
- Empty elements are **closed in the start tag**
- Most HTML elements can have **attributes**

Tip: You will learn about attributes in the next chapter of this tutorial.

Nested HTML Elements
Most HTML elements can be nested (can contain other HTML elements).
HTML documents consist of nested HTML elements.

HTML Document Example
<html>
<body>
* This is my first paragraph. *

</body>
</html>

The example above contains 3 HTML elements.

HTML Example Explained
The element:
This is my first paragraph.
The element defines a paragraph in the HTML document.
The element has a start tag and an end tag .
The element content is: This is my first paragraph.
The <body> element:
<body>
This is my first paragraph.
</body>
The <body> element defines the body of the HTML document.
The element has a start tag <body> and an end tag </body>.
The element content is another HTML element (a p element).
The <html> element:
<html>

<body>
This is my first paragraph.
</body>

</html>
The <html> element defines the whole HTML document.
The element has a start tag <html> and an end tag </html>.
The element content is another HTML element (the body element).

Don't Forget the End Tag
Some HTML elements might display correctly even if you forget the end tag:
This is a paragraph
This is a paragraph
The example above works in most browsers, because the closing tag is considered optional.
Never rely on this. Many HTML elements will produce unexpected results and/or errors if you forget the end tag .

Empty HTML Elements
HTML elements with no content are called empty elements.

 is an empty element without a closing tag (the
 tag defines a line break).
Tip: In XHTML, all elements must be closed. Adding a slash inside the start tag, like
, is the proper way of closing empty elements in XHTML (and XML).

HTML Tip: Use Lowercase Tags
HTML tags are not case sensitive: <P> means the same as . Many web sites use uppercase HTML tags.
W3Schools use lowercase tags because the World Wide Web Consortium (W3C) **recommends** lowercase in HTML 4, and **demands** lowercase tags in XHTML.

HTML Attributes

- HTML elements can have **attributes**
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**

- Attributes come in name/value pairs like: **name="value"**

Attribute Example
HTML links are defined with the <a> tag. The link address is specified in the **href attribute**:
Example
This is a link

Always Quote Attribute Values
Attribute values should always be enclosed in quotes.
Double style quotes are the most common, but single style quotes are also allowed.
Tip: In some rare situations, when the attribute value itself contains quotes, it is necessary to use single quotes: name='John "ShotGun" Nelson'

HTML Tip: Use Lowercase Attributes
Attribute names and attribute values are case-insensitive.
However, the World Wide Web Consortium (W3C) recommends lowercase attributes/attribute values in their HTML 4 recommendation.
Newer versions of (X)HTML will demand lowercase attributes.

HTML Attributes Reference
Below is a list of some attributes that are standard for most HTML elements:

Attribute	Value	Description
class	*classname*	Specifies a classname for an element
id	*id*	Specifies a unique id for an element
style	*style_definition*	Specifies an inline style for an element
title	*tooltip_text*	Specifies extra information about an element (displayed as a tool tip)

HTML Headings
Headings are defined with the <h1> to <h6> tags.
<h1> defines the most important heading. <h6> defines the least important heading.
Example
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
Note: Browsers automatically add some empty space (a margin) before and after each heading.

Headings Are Important
Use HTML headings for headings only. Don't use headings to make text **BIG** or **bold**.
Search engines use your headings to index the structure and content of your web pages.
Since users may skim your pages by its headings, it is important to use headings to show the document structure.
H1 headings should be used as main headings, followed by H2 headings, then the less important H3 headings, and so on.

HTML Lines
The <hr /> tag creates a horizontal line in an HTML page.
The hr element can be used to separate content:
Example
This is a paragraph
<hr />

This is a paragraph
<hr />
This is a paragraph

HTML Comments

Comments can be inserted into the HTML code to make it more readable and understandable. Comments are ignored by the browser and are not displayed.
Comments are written like this:

Example

<!-- This is a comment -->

Note: There is an exclamation point after the opening bracket, but not before the closing bracket.

HTML Tip - How to View HTML Source

Have you ever seen a Web page and wondered "Hey! How did they do that?"
To find out, right-click in the page and select "View Source" (IE) or "View Page Source" (Firefox), or similar for other browsers. This will open a window containing the HTML code of the page.

HTML Tag Reference

You will learn more about HTML tags and attributes in the next chapters of this tutorial.

Tag	Description
<html>	Defines an HTML document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<hr />	Defines a horizontal line
<!-->	Defines a comment

HTML Paragraphs

Paragraphs are defined with the tag.

Example

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

Note: Browsers automatically add an empty line before and after a paragraph.

Don't Forget the End Tag

Most browsers will display HTML correctly even if you forget the end tag:

Example

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

The example above will work in most browsers, but don't rely on it. Forgetting the end tag can produce unexpected results or errors.
Note: Future version of HTML will not allow you to skip end tags.

HTML Line Breaks

Use the
 tag if you want a line break (a new line) without starting a new paragraph:

Example

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

The
 element is an empty HTML element. It has no end tag.

---

 or

In XHTML, XML, elements with no end tag (closing tag) are not allowed.

Even if
 works in all browsers, writing
 instead works better in XHTML and XML applications.

HTML Text Formatting

This text is bold

This text is big

This text is italic

This is computer output

This is $_{\text{subscript}}$ and $^{\text{superscript}}$

HTML Formatting Tags

HTML uses tags like <b> and <i> for formatting output, like **bold** or *italic* text.
These HTML tags are called formatting tags (look at the bottom of this page for a complete reference).

Often renders as <b>, and renders as <i>.

However, there is a difference in the meaning of these tags:

<b> or <i> defines bold or italic text only.

 or means that you want the text to be rendered in a way that the user understands as "important". Today, all major browsers render strong as bold and em as italics. However, if a browser one day wants to make a text highlighted with the strong feature, it might be cursive for example and not bold!

HTML Text Formatting Tags

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

HTML Fonts

The HTML <font> Tag Should NOT be Used
The <font> tag is deprecated in HTML 4, and removed from HTML5.
The World Wide Web Consortium (W3C) has removed the <font> tag from its recommendations.
In HTML 4, style sheets (CSS) should be used to define the layout and display properties for many HTML elements.

The example below shows how the HTML could look by using the <font> tag:

Example

```
<p>
<font size="5" face="arial" color="red">
This paragraph is in Arial, size 5, and in red text color.
</font>
</p>

<p>
<font size="3" face="verdana" color="blue">
This paragraph is in Verdana, size 3, and in blue text color.
</font>
</p>
```

Styling HTML with CSS

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
  body {background-color:lightgray}
  h1   {color:blue}
  p    {color:green}
</style>
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

CSS stands for **C**ascading **S**tyle **S**heets

Styling can be added to HTML elements in 3 ways:

- Inline - using a **style attribute** in HTML elements
- Internal - using a **<style> element** in the HTML <head> section
- External - using one or more **external CSS files**

The most common way to add styling, is to keep the CSS syntax in separate CSS files. But, in this tutorial, the examples are internal, and simplified, to make it easier for you understand it, and try it yourself.

CSS Syntax

CSS styling has the following **syntax**:

element { property:value ; propety:value }

The **element** is an HTML element name. The ***property*** is a CSS property. The ***value*** is a CSS value.

Multiple styles are separated with semicolon.

Inline Styling (Inline CSS)

Inline styling is useful for applying a unique style to a single HTML element:

Inline styling uses the **style attribute**.

This inline styling changes the text color of a single paragraph:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1 style="color:blue">This is a Blue Heading</h1>
</body>
</html>
```

Internal Styling (Internal CSS)

An internal style sheet can be used to define a common style for all HTML elements on a page.

Internal styling is defined in the **<head>** section of an HTML page, using a **<style>** element:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
 body {background-color:lightgrey}
 h1   {color:blue}
 p    {color:green}
</style>
</head>
<body>
 <h1>This is a heading</h1>
 <p>This is a paragraph.</p>
</body>
</html>
```

External Styling (External CSS)

External style sheet are ideal when the style is applied to many pages.

With external style sheets, you can change the look of an entire site by changing one file.

External styles are defined in the **<head>** section of an HTML page, in the **<link>** element:

Example

```
<DOCTYPE html>
<html>
<head>
 <link rel="stylesheet" href="styles.css">
</head>
<body>
 <h1>This is a heading</h1>
 <p>This is a paragraph.</p>
</body>
</html>
```

CSS Fonts

The CSS property **color** defines the text color to be used for an HTML element.

The CSS property **font-family** defines the font to be used for an HTML element.

The CSS property **font-size** defines the text size to be used for an HTML element.

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
   color:blue;
   font-family:verdana;
   font-size:300%;
}
p  {
   color:red;
   font-family:courier;
   font-size:160%
}
</style>
</head>
<html>
<body>
```

```
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

The CSS Box Model

Every visible HTML element has a box around it, even if you cannot see it.

The CSS **border** property defines a visible border around an HTML element:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
   border:1px solid grey;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```

The CSS **padding** property defines a padding (space) inside the border:

Example

```
 <!DOCTYPE html>
<html>
<head>
<style>
p {
   border:1px solid grey;
   padding:10px;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```

The CSS **margin** property defines a margin (space) outside the border:

Example

```
 <!DOCTYPE html>
<html>
<head>
<style>
p {
   border:1px solid grey;
   padding:10px;
   margin:30px;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```

The CSS examples above use px to define sizes in pixels (screen pixels)

The id Attribute

All the examples above use CSS to style HTML elements in a general way.
The CSS styles define an equal style for all equal elements.
To define a special style for a special element, first add an id attribute to the element:

Example

I am different

then define a different style for the (identified) element:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
p#p01 {
    color: blue;
}
</style>
</head>
<body>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p id="p01">I am different.</p>
</body>
</html>
```

CSS full demo

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

<div style="position:relative;">
  <div style="opacity:0.5;position:absolute;left:50px;top:-30px;width:300px;height:150px;background-color:#40B3DF"></div>
  <div style="opacity:0.3;position:absolute;left:120px;top:20px;width:100px;height:170px;background-color:#8AC007"></div>
  <div style="margin-top:30px;width:360px;height:130px;padding:20px;border-radius:10px;border:10px solid #EE872A;font-size:120%;">
            <h1>CSS = Styles and Colors</h1>
            <div style="letter-spacing:12px;font-size:15px;position:relative;left:25px;top:25px;">Manipulate Text</div>
            <div style="color:#40B3DF;letter-spacing:12px;font-size:15px;position:relative;left:25px;top:30px;">Colors,
            <span style="background-color:#B4009E;color:#ffffff;"> Boxes</span></div>
            </div>
</div>

</body>
</html>
```

HTML LINKS

Links are found in nearly all Web pages. Links allow users to click their way from page to page.

HTML Hyperlinks (Links)

A hyperlink (or link) is a word, group of words, or image that you can click on to jump to a new document or a new section within the current document.
When you move the cursor over a link in a Web page, the arrow will turn into a little hand.
Links are specified in HTML using the <a> tag.
The <a> tag can be used in two ways:

1. To create a link to another document, by using the href attribute
2. To create a bookmark inside a document, by using the name attribute

---

HTML Link Syntax

The HTML code for a link is simple. It looks like this:
Link text
The href attribute specifies the destination of a link.
Example
Visit W3Schools
which will display like this: Visit W3Schools
Clicking on this hyperlink will send the user to W3Schools' homepage.

Tip: The "*Link text*" doesn't have to be text. It can be an image or any other HTML element.

Local Links
The example above used an absolute URL (A full web address).
A local link (link to the same web site) is specified with a relative URL (without http://www....).
Example:
HTML Images

HTML Links - Colors and Icons
When you move the mouse cursor over a link, two things will normally happen:

- The mouse arrow will turn into a little hand
- The color of the link element will change

By default, links will appear as this in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

You can change the default colors, using styles:
Example

```
<!DOCTYPE html>
<html>
<head>
<style>
a:link    {color:#000000; background-color:transparent}
a:visited {color:#000000; background-color:transparent}
a:hover   {color:#ff0000; background-color:transparent}
a:active  {color:#ff0000; background-color:transparent}
</style>
</head>
<body>
<p>You can change the default colors of links</p>
<a href="html_images.asp" target="_blank">HTML Images</a>
</body>
</html>
```

HTML Links - The target Attribute
The target attribute specifies where to open the linked document.
The example below will open the linked document in a new browser window or a new tab:
Example
Visit W3Schools!

Target Value	Description
_blank	Opens the linked document in a new window or tab
_self	Opens the linked document in the same frame as it was clicked (this is default)
_parent	Opens the linked document in the parent frame
_top	Opens the linked document in the full body of the window
framename	Opens the linked document in a named frame

If your webpage is locked in a frame, you can use target="_top" to break out of the frame:
Example
HTML5 tutorial!

HTML Links - The name Attribute
The name attribute specifies the name of an anchor.
The name attribute is used to create a bookmark inside an HTML document.

Note: The upcoming HTML5 standard suggests using the id attribute instead of the name attribute for specifying the name of an anchor. Using the id attribute actually works also for HTML4 in all modern browsers.
Bookmarks are not displayed in any special way. They are invisible to the reader.
Example
A named anchor inside an HTML document:
Useful Tips Section
Create a link to the "Useful Tips Section" inside the same document:
Visit the Useful Tips Section
Or, create a link to the "Useful Tips Section" from another page:

Visit the Useful Tips Section

HTML Links - Image as Link
It is common to use images as links:
Example

```
<a href="default.asp">
  <img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;border:0">
</a>
```

Basic Notes - Useful Tips
Tip: 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.

HTML Links - The id Attribute
The **id** attribute can be used to create bookmarks inside HTML documents.
Bookmarks are not displayed in any special way. They are invisible to the reader.
Example
Add an id attribute to any <a> element:
Useful Tips Section
Then create a link to the <a> element (Useful Tips Section):
Visit the Useful Tips Section
Or, create a link to the <a> element (Useful Tips Section) from another page:
Visit the Useful Tips Section

HTML Link Tags

Tag	Description
<a>	Defines an anchor

***HTML IMAGES* - The <img> 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 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.
Syntax for defining an image:

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

HTML Images - The Alt Attribute

The required alt attribute specifies an alternate text for an image, if the image cannot be displayed. The value of the alt attribute is an author-defined text:

The alt attribute provides alternative information for an image if a user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

HTML Images - Set Height and Width of an Image

The height and width attributes are used to specify the height and width of an image.

The attribute values are specified in pixels by default:

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

Basic Notes - Useful Tips

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

Note: When a web page is loaded, it is the browser, at that moment, that actually gets the image from a web server and inserts it into the page. Therefore, make sure that the images actually stay in the same spot in relation to the web page, otherwise your visitors will get a broken link icon. The broken link icon is shown if the browser cannot find the image.

HTML Image Tags

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

HTML Tables

Tables are defined with the <table> tag.

A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc.

Table Example

```
<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 the HTML code above looks in a browser:

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

HTML Tables and the Border Attribute

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

To display a table with borders, specify the border attribute:

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

HTML Table Headers

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

All major browsers display the text in the <th> element as bold and centered.

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

How the HTML code above looks in your browser:

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

HTML Table Tags

Tag	Description
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines a group of columns in a table, for formatting
<col />	Defines attribute values for one or more columns in a table
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table

Table Summary

- Use the HTML **<table>** element to define a table
- Use the HTML **<tr>** element to define a table row
- Use the HTML **<td>** element to define a table data
- Use the HTML **<th>** element to define a table heading

- Use the HTML **<caption>** element to define a table caption
- Use the CSS **border** property to define a border
- Use the CSS **border-collapse** property to collapse cell borders
- Use the CSS **padding** property to add padding to cells
- Use the CSS **text-align** property to align cell text
- Use the CSS **border-spacing** property to set the spacing between cells
- Use the **colspan** attribute to make a cell span many columns
- Use the **rowspan** attribute to make a cell span many rows
- Use the **id** attribute to uniquely define one table

HTML Lists

The most common HTML lists are ordered and unordered lists:

An ordered list:

1. The first list item
2. The second list item
3. The third list item

An unordered list:

- List item
- List item
- List item

HTML Unordered Lists

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.
The list items are marked with bullets (typically small black circles).

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

How the HTML code above looks in a browser:

- Coffee
- Milk
- *Unordered HTML Lists - The Style Attribute*
- A **style** attribute can be added to an **unordered list**, to define the style of the marker:

Style	Description
list-style-type:disc	The list items will be marked with bullets (default)
list-style-type:circle	The list items will be marked with circles
list-style-type:square	The list items will be marked with squares
list-style-type:none	The list items will not be marked

Disc:

```
<!DOCTYPE html>
<html>
<body>
<h2>Unordered List with Disc Bullets</h2>
<ul style="list-style-type:disc">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>
</body>
</html>
```

Circle:

```
<!DOCTYPE html>
<html>
<body>
<h2>Unordered List with Circle Bullets</h2>
<ul style="list-style-type:circle">
```

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

Square:

```
<!DOCTYPE html>
<html>
<body>
<h2>Unordered List with Square Bullets</h2>
<ul style="list-style-type:square">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>
</body>
</html>
```

None:

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

<h2>Unordered List without Bullets</h2>
<ul style="list-style-type:none">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>
</body>
</html>
```

HTML Ordered Lists

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.
The list items are marked with numbers.

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

How the HTML code above looks in a browser:

1. Coffee
2. Milk

Ordered HTML Lists - The Type Attribute

A **type** attribute can be added to an **ordered list**, to define the type of the marker:

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

Numbers:

```
<ol type="1">
  <li>Coffee</li>
  <li>Tea
```

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

Upper Case:

```
<ol type="A">
  <li>Coffee</li>
  <li>Tea
  <li>Milk</li>
</ol>
```

Lower Case:

```
<ol type="a">
  <li>Coffee</li>
  <li>Tea
  <li>Milk</li>
</ol>
```

Roman Upper Case:

```
<ol type="I">
  <li>Coffee</li>
  <li>Tea
  <li>Milk</li>
</ol>
```

Roman Lower Case:

```
<ol type="i">
  <li>Coffee</li>
  <li>Tea
  <li>Milk</li>
</ol>
```

HTML Definition Lists

A definition list is a list of items, with a description of each item.

The <dl> tag defines a definition list.

The <dl> tag is used in conjunction with <dt> (defines the item in the list) and <dd> (describes the item in the list):

```
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dt>Milk</dt>
<dd>- white cold drink</dd>
</dl>
```

How the HTML code above looks in a browser:

Coffee

- black hot drink

Milk

- white cold drink

Nested HTML Lists

List can be nested (lists inside lists).

Nested Lists:

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Output:

A Nested List

- Coffee
- Tea
  - Black tea
  - Green tea
- Milk

Horizontal Lists

HTML lists can be styled in many different ways with CSS.

One popular way, is to style a list to display horizontally:

Horizontal List:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul#menu li {
    display:inline;
}
</style>
</head>
<body>
<h2>Horizontal List</h2>
<ul id="menu">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>

</body>
</html>
```

With a little extra style, you can make it look like a menu:

Tables Lists Blocks Classes

New Style:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul#menu {
    padding: 0;
}
ul#menu li {
    display: inline;
}
ul#menu li a {
    background-color: black;
    color: white;
    padding: 10px 20px;
    text-decoration: none;
    border-radius: 4px 4px 0 0;
}
ul#menu li a:hover {
    background-color: red;
}
</style>
</head>
<body>
<h2>Horizontal List</h2>
<ul id="menu">
 <li><a href="html_tables.asp">Tables</a></li>
 <li><a href="html_lists.asp">Lists</a></li>
 <li><a href="html_blocks.asp">Blocks</a></li>
 <li><a href="html_classes.asp">Classes</a></li>
</ul>
</body>
</html>
```

HTML List Tags

Tag	Description
	Defines an ordered list
	Defines an unordered list

	Defines a list item
<dl>	Defines a definition list
<dt>	Defines an item in a definition list
<dd>	Defines a description of an item in a definition list

HTML* Lists *Summary

- Use the HTML **<ul>** element to define an unordered list
- Use the HTML **style** attribute to define the bullet style
- Use the HTML **<ol>** element to define an ordered list
- Use the HTML **type** attribute to define the numbering type
- Use the HTML **<li>** element to define a list item
- Use the HTML **<dl>** element to define a description list
- Use the HTML **<dt>** element to define the description term
- Use the HTML **<dd>** element to define the description data
- Lists can be nested inside lists
- List items can contain other HTML elements
- Use the CSS property **display:inline** to display a list horizontally

HTML Block Elements

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.
Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.

Example

```
<!DOCTYPE html>
<html>
<body>
<div style="background-color:black; color:white; margin:20px; padding:20px;">
<h2>London</h2>
<p>
London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.
</p>
<p>
Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.
</p>
</div>
</body>
</html>
```

HTML Block Elements and Inline Elements

Most HTML elements are defined as **block level** elements or **inline** elements.
Block level elements normally start (and end) with a new line, when displayed in a browser.
Examples: <h1>, <p>, <ul>, <table>
Inline elements are normally displayed without line breaks.
Examples: <b>, <td>, <a>, <img>

The HTML Element

The HTML element is a **block level element** that can be used as a container for other HTML elements.
The element has no special meaning. It has no required attributes, but **style** and **class** are common.
Because it is a block level element, the browser will display line breaks before and after it.

When used together with CSS, the element can be used to style blocks of content.

The HTML Element

The HTML element is an **inline element** that can be used as a container for text.
The element has no special meaning. It has no required attributes, but **style** and **class** are common.
Unlike , which is formatted with line breaks, the element does not have any automatic formatting.
When used together with CSS, the element can be used to style parts of the text:
Example

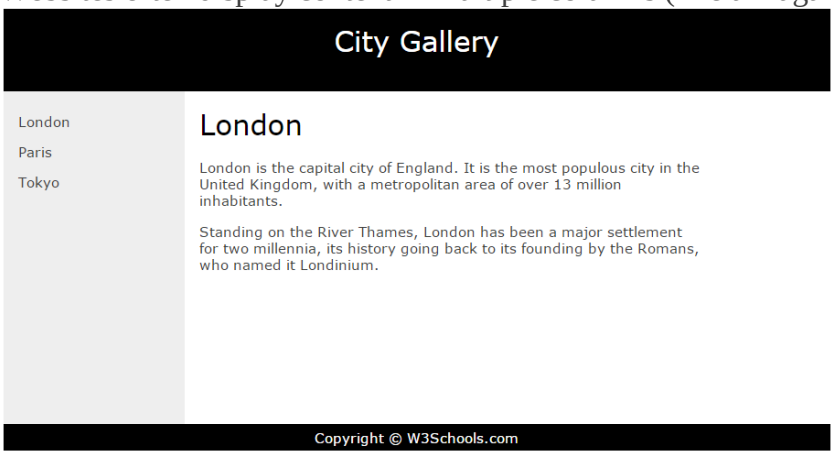
```
<!DOCTYPE html>
<html>
<body>
<h1>My <span style="color:red">Important</span> Heading</h1>
</body>
</html>
```

HTML Grouping Tags

Tag	Description
	Defines a section in a document (block-level)
	Defines a section in a document (inline)

HTML Layouts

Websites often display content in multiple columns (like a magazine or newspaper).



HTML Layout Using Elements

The element is often used as a layout tool, because it can easily be positioned with CSS.

This example uses 4 elements to create a multiple column layout:
Example

```
<!DOCTYPE html>
<html>
<head>
<style>
#header {
    background-color:black;
    color:white;
    text-align:center;
    padding:5px;
}
#nav {
    line-height:30px;
    background-color:#eeeeee;
    height:300px;
    width:100px;
    float:left;
```

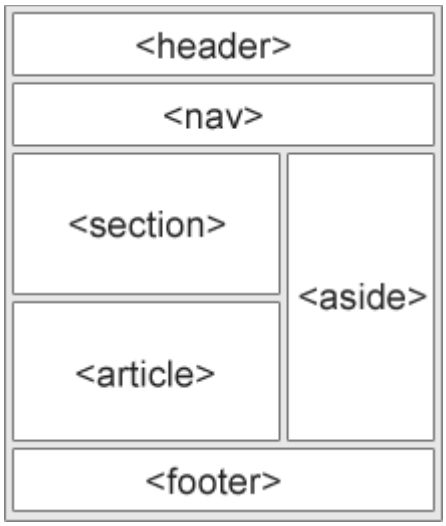
```
    padding:5px;
}
#section {
    width:350px;
    float:left;
    padding:10px;
}
#footer {
    background-color:black;
    color:white;
    clear:both;
    text-align:center;
   padding:5px;
}
</style>
</head>
<body>
<div id="header">
<h1>City Gallery</h1>
</div>
<div id="nav">
London<br>
Paris<br>
Tokyo<br>
</div>
<div id="section">
<h2>London</h2>
<p>
London is the capital city of England. It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.
</p>
<p>
Standing on the River Thames, London has been a major settlement for two millennia,
its history going back to its founding by the Romans, who named it Londinium.
</p>
</div>
<div id="footer">
Copyright © W3Schools.com
</div>
</body>
</html>
```

The CSS:

```
<style>
#header {
    background-color:black;
    color:white;
    text-align:center;
    padding:5px;
}
#nav {
    line-height:30px;
    background-color:#eeeeee;
    height:300px;
    width:100px;
    float:left;
    padding:5px;
}
#section {
    width:350px;
    float:left;
    padding:10px;
}
#footer {
    background-color:black;
    color:white;
    clear:both;
    text-align:center;
    padding:5px;
}
</style>
```

Website Layout Using HTML5

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



header	Defines a header for a document or a section
nav	Defines a container for navigation links
section	Defines a section in a document
article	Defines an independent self-contained article
aside	Defines content aside from the content (like a sidebar)
footer	Defines a footer for a document or a section
details	Defines additional details
summary	Defines a heading for the details element

```
<!DOCTYPE html>
<html>

<head>
<style>
header {
    background-color:black;
    color:white;
    text-align:center;
    padding:5px;
}
nav {
    line-height:30px;
    background-color:#eeeeee;
    height:300px;
    width:100px;
    float:left;
    padding:5px;
}
section {
    width:350px;
    float:left;
    padding:10px;
}
footer {
    background-color:black;
    color:white;
    clear:both;
    text-align:center;
    padding:5px;
}
</style>
</head>

<body>

<header>
<h1>City Gallery</h1>
</header>

<nav>
London<br>
Paris<br>
Tokyo<br>
</nav>

<section>
<h1>London</h1>
<p>
London is the capital city of England. It is the most populous city in the United Kingdom,
```

```
with a metropolitan area of over 13 million inhabitants.
</p>
<p>
Standing on the River Thames, London has been a major settlement for two millennia,
its history going back to its founding by the Romans, who named it Londinium.
</p>
</section>

<footer>
Copyright © W3Schools.com
</footer>

</body>
</html>
```

The CSS

```
<style>
header {
    background-color:black;
    color:white;
    text-align:center;
    padding:5px;
}
nav {
    line-height:30px;
    background-color:#eeeeee;
    height:300px;
    width:100px;
    float:left;
    padding:5px;
}
section {
    width:350px;
    float:left;
    padding:10px;
}
footer {
    background-color:black;
    color:white;
    clear:both;
    text-align:center;
    padding:5px;
}
```

HTML Forms and Input

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

Create text fields

How to create text fields. The user can write text in a text field.

```
<!DOCTYPE html>
<html>
<body>
<form action="">
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname">
</form>
<p><b>Note:</b> The form itself is not visible. Also note that the default width of a text field is 20 characters.</p>
</body>
</html>
```

Create password field

How to create a password field.

```
<!DOCTYPE html>
<html>
<body>
<form action="">
Username: <input type="text" name="user"><br>
Password: <input type="password" name="password">
</form>
<p><b>Note:</b> The characters in a password field are masked (shown as asterisks or circles).</p>
</body>
</html>
```

HTML Forms

HTML forms are used to pass data to a server.

An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

The <form> tag is used to create an HTML form:

```
<form>
.
input elements
.
</form>
```

HTML Forms - The Input Element

The most important form element is the <input> element.

The <input> element is used to select user information.

An <input> element can vary in many ways, depending on the type attribute. An <input> element can be of type text field, checkbox, password, radio button, submit button, and more.

The most common input types are described below.

Text Fields

<input type="text"> defines a one-line input field that a user can enter text into:

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

How the HTML code above looks in a browser:

First name:

Last name:

Note: The form itself is not visible. Also note that the default width of a text field is 20 characters.

Password Field

<input type="password"> defines a password field:

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

How the HTML code above looks in a browser:

Password:

Note: The characters in a password field are masked (shown as asterisks or circles).

Radio Buttons

<input type="radio"> defines a radio button. Radio buttons let a user select ONLY 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 the HTML code above looks in a browser:

Male

Female

Checkboxes

<input type="checkbox"> defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
```

```
<input type="checkbox" name="vehicle" value="Car">I have a car
</form>
```
How the HTML code above looks in a browser:

I have a bike

I have a car

Submit Button

<input type="submit"> defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="demo_form_action.asp" method="get">
Username: <input type="text" name="user">
<input type="submit" value="Submit">
</form>
```
How the HTML code above looks in a browser:

Username: Submit

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

More Examples

Simple drop-down list

How to create a simple drop-down list.

```
<!DOCTYPE html>
<html>
<body>
<form action="">
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
</form>
</body>
</html>
```

Drop-down list with a pre-selected value

How to create a drop-down list with a pre-selected value.

```
<!DOCTYPE html>
<html>
<body>
<form action="">
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat" selected>Fiat</option>
<option value="audi">Audi</option>
</select>
</form>
</body>
</html>
```

Textarea

How to create a multi-line text input control. In a text-area the user can write an unlimited number of characters.

```
<!DOCTYPE html>
<html>
<body>
<textarea rows="10" cols="30">
The cat was playing in the garden.
</textarea>
</body>
```

```
</html>
```

Create a button

How to create a button.

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

Form Examples

Fieldset around form-data

How to create a border around elements in a form.

```
<!DOCTYPE html>
<html>
<body>
<form action="">
<fieldset>
<legend>Personal information:</legend>
Name: <input type="text" size="30"><br>
E-mail: <input type="text" size="30"><br>
Date of birth: <input type="text" size="10">
</fieldset>
</form>
</body>
</html>
```

Form with text fields and a submit button

How to create a form with two text fields and a submit button.

```
<!DOCTYPE html>
<html>
<body>
<form name="input" action="demo_form_action.asp" method="get">
First name: <input type="text" name="FirstName" value="Mickey"><br>
Last name: <input type="text" name="LastName" value="Mouse"><br>
<input type="submit" value="Submit">
</form>
<p>If you click the "Submit" button, the form-data will be sent to a page called "demo_form_action.asp".</p>
</body>
</html>
```

Send e-mail from a form

How to send e-mail from a form.

```
<!DOCTYPE html>
<html>
<body>
<h2>Send e-mail to someone@example.com:</h2>
<form action="MAILTO:someone@example.com" method="post" enctype="text/plain">
Name:<br>
<input type="text" name="name" value="your name"><br>
E-mail:<br>
<input type="text" name="mail" value="your email"><br>
Comment:<br>
<input type="text" name="comment" value="your comment" size="50"><br><br>
<input type="submit" value="Send">
<input type="reset" value="Reset">
</form>
</body>
</html>
```

Send e-mail to someone@example.com:

Name:
your name
E-mail:
your email
Comment:
your comment

Send Reset

HTML Form Tags

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<button>	Defines a clickable button
<datalist>	Specifies a list of pre-defined options for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

HTML IFRAMES

An iframe is used to display a web page within a web page.

Iframe Syntax

The syntax for adding an iframe is:

<iframe src="*URL*"></iframe>

The **src** attribute specifies the URL (web address) of the iframe page.

Iframe - Set Height and Width

Use the **height** and **width** attributes to specify the size.

The attribute values are specified in pixels by default, but they can also be in percent (like "80%").

Example

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

<iframe src="demo_iframe.htm" width="200" height="200"></iframe>
```

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

Iframe - Remove the Border

The **frameborder** attribute specifies whether or not to display a border around the iframe.
Set the attribute value to "0" to remove the border:

Example

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

<iframe src="demo_iframe.htm" frameborder="0"></iframe>

</body>
</html>
```

Use iframe as a Target for a Link

An iframe can be used as the target frame for a link.
The **target** attribute of the link must refer to the **name** attribute of the iframe:

Example

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

<iframe width="100%" height="300px" src="demo_iframe.htm" name="iframe_a"></iframe>
<p><a href="http://www.w3schools.com" target="iframe_a">W3Schools.com</a></p>

<p>When the target of a link matches the name of an iframe, the link will open in the iframe.</p>

</body>
</html>
```

HTML iframe Tag

Tag	Description
<iframe>	Defines an inline frame

DHTML

DHTML Introduction

DHTML is NOT a Language
DHTML stands for **D**ynamic **HTML**.
DHTML is NOT a language or a web standard.
To most people DHTML means the combination of HTML, JavaScript, DOM and CSS.
According to the World Wide Web Consortium (W3C):
"Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."

JAVASCRIPT

JavaScript Introduction

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.

What You Should Already Know

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

- HTML and CSS

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Are Java and JavaScript the same?

NO!

Java and JavaScript are two completely different languages in both concept and design!

Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What Can 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 react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser -** A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies -** A JavaScript can be used to store and retrieve information on the visitor's computer

---

JavaScript How To

The HTML <script> tag is used to insert a JavaScript into an HTML page.

Writing to The HTML Document

The example below writes a <p> element with current date information to the HTML document:

Example

```
<html>
<body>
<h1>My First Web Page</h1>
<script type="text/javascript">
document.write("<p>" + Date() + "</p>");
</script>
</body>
</html>
```

Note: Try to avoid using document.write() in real life JavaScript code. The entire HTML page will be overwritten if document.write() is used inside a function, or after the page is loaded. However, document.write() is an easy way to demonstrate JavaScript output in a tutorial.

Changing HTML Elements

The example below writes the current date into an existing element:

Example

```
<html>
<body>

<h1>My First Web Page</h1>

<p id="demo"></p>

<script type="text/javascript">
document.getElementById("demo").innerHTML=Date();
</script>

</body>
</html>
```

Note: To manipulate HTML elements JavaScript uses the DOM method **getElementById()**. This method accesses the element with the specified id.

Examples Explained

To insert a JavaScript into an HTML page, use the <script> tag.

Inside the <script> tag use the type attribute to define the scripting language.

The <script> and </script> tells where the JavaScript starts and ends:

<html>
<body>
<h1>My First Web Page</h1>

* This is a paragraph. *

<script type="text/javascript">
... some JavaScript code ...
</script>

</body>
</html>

The lines between the <script> and </script> contain the JavaScript and are executed by the browser.

In this case the browser will replace the content of the HTML element with id="demo", with the current date:

<html>
<body>
<h1>My First Web Page</h1>

* This is a paragraph. *

<script type="text/javascript">
document.getElementById("demo").innerHTML=Date();
</script>

</body>
</html>

Without the <script> tag(s), the browser will treat "document.getElementById("demo").innerHTML=Date();" as pure text and just write it to the page: Try it yourself

---

Some Browsers do Not Support JavaScript
Browsers that do not support JavaScript, will display JavaScript as page content.
To prevent them from doing this, and as a part of the JavaScript standard, the HTML comment tag should be used to "hide" the JavaScript.
Just add an HTML comment tag <!-- before the first JavaScript statement, and a --> (end of comment) after the last JavaScript statement, like this:

```
<html>
<body>
<script type="text/javascript">
<!--
document.getElementById("demo").innerHTML=Date();
//-->
</script>
</body>
</html>
```

The two forward slashes at the end of comment line (//) is the JavaScript comment symbol. This prevents JavaScript from executing the --> tag.

JavaScript Where To
JavaScripts can be put in the <body> and in the <head> sections of an HTML page.

JavaScript in <body>
The example below writes the current date into an existing <p> element when the page loads:
Example

```
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<script type="text/javascript">
document.getElementById("demo").innerHTML=Date();
</script>
</body>
</html>
```

Note that the JavaScript is placed at the bottom of the page to make sure it is not executed before the <p> element is created.

JavaScript Functions and Events
JavaScripts in an HTML page will be executed when the page loads. This is not always what we want.
Sometimes we want to execute a JavaScript when an **event** occurs, such as when a user clicks a button. When this is the case we can put the script inside a **function**.
Events are normally used in combination with functions (like calling a function when an event occurs).
You will learn more about JavaScript functions and events in later chapters.

JavaScript in <head>
The example below calls a function when a button is clicked:

Example

```
<html>
<head>
<script type="text/javascript">
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<button type="button" onclick="displayDate()">Display Date</button>
</body>
</html>
```

Scripts in <head> and <body>

You can place an unlimited number of scripts in your document, and you can have scripts in both the body and the head section at the same time.

It is a common practice to put all functions in the head section, or at the bottom of the page. This way they are all in one place and do not interfere with page content.

Using an External JavaScript

JavaScript can also be placed in external files.

External JavaScript files often contain code to be used on several different web pages.

External JavaScript files have the file extension .js.

Note: External script cannot contain the <script></script> tags!

To use an external script, point to the .js file in the "src" attribute of the <script> tag:

Example

```
<html>
<head>
<script type="text/javascript" src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

Note: Remember to place the script exactly where you normally would write the script!

JavaScript Statements

JavaScript is a sequence of statements to be executed by the browser.

JavaScript is Case Sensitive

Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

JavaScript Statements

A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do.

This JavaScript statement tells the browser to write "Hello Dolly" to the web page:

document.write("Hello Dolly");

It is normal to add a semicolon at the end of each executable statement. Most people think this is a good programming practice, and most often you will see this in JavaScript examples on the web.

The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Because of this you will often see examples without the semicolon at the end.
Note: Using semicolons makes it possible to write multiple statements on one line.

JavaScript Code
JavaScript code (or just JavaScript) is a sequence of JavaScript statements.
Each statement is executed by the browser in the sequence they are written.
This example will write a heading and two paragraphs to a web page:
Example

```
<script type="text/javascript">
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Blocks
JavaScript statements can be grouped together in blocks.
Blocks start with a left curly bracket {, and end with a right curly bracket }.
The purpose of a block is to make the sequence of statements execute together.
This example will write a heading and two paragraphs to a web page:
Example

```
<script type="text/javascript">
{
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
}
</script>
```

The example above is not very useful. It just demonstrates the use of a block. Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).
You will learn more about functions and conditions in later chapters.

JavaScript Comments
JavaScript comments can be used to make the code more readable.

JavaScript Comments
Comments can be added to explain the JavaScript, or to make the code more readable.
Single line comments start with //.
The following example uses single line comments to explain the code:
Example

```
<script type="text/javascript">
// Write a heading
document.write("<h1>This is a heading</h1>");
// Write two paragraphs:
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Multi-Line Comments
Multi line comments start with /* and end with */.
The following example uses a multi line comment to explain the code:

Example

```
<script type="text/javascript">
/*
The code below will write
one heading and two paragraphs
*/
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

Using Comments to Prevent Execution

In the following example the comment is used to prevent the execution of a single code line (can be suitable for debugging):

Example

```
<script type="text/javascript">
//document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

In the following example the comment is used to prevent the execution of a code block (can be suitable for debugging):

Example

```
<script type="text/javascript">
/*
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
*/
</script>
```

Using Comments at the End of a Line

In the following example the comment is placed at the end of a code line:

Example

```
<script type="text/javascript">
document.write("Hello"); // Write "Hello"
document.write(" Dolly!"); // Write " Dolly!"
</script>
```

JavaScript Variables

Variables are "containers" for storing information.

Do You Remember Algebra From School?

Do you remember algebra from school? x=5, y=6, z=x+y

Do you remember that a letter (like x) could be used to hold a value (like 5), and that you could use the information above to calculate the value of z to be 11?

These letters are called **variables**, and variables can be used to hold values (x=5) or expressions (z=x+y).

JavaScript Variables

As with algebra, JavaScript variables are used to hold values or expressions.

A variable can have a short name, like x, or a more descriptive name, like carname.

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)

- Variable names must begin with a letter, the $ character, or the underscore character

Note: Because JavaScript is case-sensitive, variable names are case-sensitive.

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.
You declare JavaScript variables with the **var** keyword:

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).
However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.
Note: When you assign a text value to a variable, put quotes around the value.
Note: If you redeclare a JavaScript variable, it will not lose its value.

Local JavaScript Variables

A variable declared within a JavaScript function becomes **LOCAL** and can only be accessed within that function. (the variable has local scope).
You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
Local variables are deleted as soon as the function is completed.
You will learn more about functions in a later chapter of this tutorial.

Global JavaScript Variables

Variables declared outside a function become **GLOBAL**, and all scripts and functions on the web page can access it.
Global variables are deleted when you close the page.

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that have not yet been declared, the variables will automatically be declared as global variables.
These statements:

```
x=5;
carname="Volvo";
```

will declare the variables x and carname as global variables (if they don't already exist).

JavaScript Arithmetic

As with algebra, you can do arithmetic operations with JavaScript variables:

```
y=x-5;
z=y+5;
```

JavaScript Operators

= is used to assign values.
+ is used to add values.

The assignment operator = is used to assign values to JavaScript variables.
The arithmetic operator + is used to add values together.

```
y=5;
z=2;
x=y+z;
```

The value of x, after the execution of the statements above, is 7.

JavaScript Arithmetic Operators
Arithmetic operators are used to perform arithmetic between variables and/or values.
Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5
-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5
++	Increment	x=++y	x=6	y=6
		x=y++	x=5	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4

JavaScript Assignment Operators
Assignment operators are used to assign values to JavaScript variables.
Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings
The + operator can also be used to add string variables or text values together.
To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".
To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:
"What a very nice day"

Adding Strings and Numbers
The rule is: **If you add a number and a string, the result will be a string!**

Example

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false x==5 is true
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

How Can it be Used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

if (age<18) document.write("Too young");

You will learn more about the use of conditional statements in the next chapter of this tutorial.

Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator
JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.
Syntax
variablename=(condition)?value1:value2
Example
Example
If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President " else it will be assigned "Dear":

```
<script type="text/javascript">

var visitor="PRES";
var greeting=(visitor=="PRES")?"Dear President ":"Dear ";
document.write(greeting);

</script>
```

JavaScript If...Else Statements
Conditional statements 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 to execute some code only if a specified condition is true
- **if...else statement** - use this statement 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 to select one of many blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

---

If Statement
Use the if statement 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

```
<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>
```

Notice that there is no ..else.. in this syntax. You tell the browser to execute some code **only if the specified condition is true**.

If...else Statement
Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.
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
Use the if....else if...else statement to select one of several blocks of code to be executed.
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 neither condition1 nor condition2 is true
  }
```

Example

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
else if (time>=10 && time<16)
  {
  document.write("<b>Good day</b>");
  }
else
  {
  document.write("<b>Hello World!</b>");
  }
</script>
```

VBScript

VBScript Introduction

What You Should Already Know

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

- HTML / XHTML

If you want to study these subjects first, find the tutorials on our Home page.

What is VBScript?

- VBScript is a scripting language
- A scripting language is a lightweight programming language
- VBScript is a light version of Microsoft's programming language Visual Basic
- **VBScript is only supported by Microsoft's browsers (Internet Explorer)**

How Does it Work?

When a VBScript is inserted into an HTML document, Internet Explorer browser will read the HTML and interpret the VBScript. The VBScript can be executed immediately, or at a later event. VBScript only works in Microsoft browsers (Internet Explorer).

VBScript How To

The HTML <script> tag is used to insert a VBScript into an HTML page.

Put a VBScript into an HTML Page

The example below shows how to use VBScript to write text on a web page:

Example (IE Only)

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

The example below shows how to add HTML tags to the VBScript:

Example (IE Only)

```
<html>
<body>
<script type="text/vbscript">
document.write("<h1>Hello World!</h1>")
</script>
</body>
</html>
```

Example Explained

To insert a VBScript into an HTML page, we use the <script> tag. Inside the <script> tag we use the type attribute to define the scripting language.

So, the <script type="text/vbscript"> and </script> tells where the VBScript starts and ends:

```
<html>
<body>
<script type="text/vbscript">
...
</script>
</body>
</html>
```

The **document.write** command is a standard VBScript command for writing output to a page.
By entering the document.write command between the <script> and </script> tags, the browser will recognize it as a VBScript command and execute the code line. In this case the browser will write Hello World! to the page:

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

How to Handle Simple Browsers

Browsers that do not support scripting, will display VBScript as page content.
To prevent them from doing this, the HTML comment tag should be used to "hide" the VBScript. Just add an HTML comment tag <!-- before the first VBScript statement, and a --> (end of comment) after the last VBScript statement, like this:

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

VBScript Where To ...

VBScripts can be placed in the body and in the head section of an HTML document.

Where to Put the VBScript

VBScripts 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, or at a later event, such as when a user clicks a button. When this is the case we put the script inside a function or a sub procedure, you will learn about procedures in a later chapter.

Scripts in <head>

Put your functions and sub procedures in the head section, this way they are all in one place, and they do not interfere with page content.

Example (IE Only)

```
<html>
<head>
<script type="text/vbscript">
function myFunction()
alert("Hello World!")
end function
</script>
</head>

<body onload="myFunction()">
</body>
</html>
```

Scripts in <body>

If you don't want your script to be placed inside a function, and especially if your script should write page content, it should be placed in the body section.

Example (IE Only)

```
<html>
<head>
</head>

<body>
<script type="text/vbscript">
document.write("This message is written by VBScript")
</script>
</body>

</html>
```

Scripts in <head> and <body>

You can place an unlimited number of scripts in your document, and you can have scripts in both the body and the head section.

Example (IE Only)

```
<html>
<head>
<script type="text/vbscript">
function myFunction()
alert("Hello World!")
end function
</script>
</head>

<body>
<button onclick="myFunction()">Click me</button>
<script type="text/vbscript">
document.write("This message is written by VBScript")
</script>
</body>
</html>
```

Using an External VBScript

If you want to run the same VBScript on several pages, without having to write the same script on every page, you can write a VBScript in an external file.

Save the external VBScript file with a .vbs file extension.

Note: The external script cannot contain the <script> tag!

To use the external script, point to the .vbs file in the "src" attribute of the <script> tag:

Example

```
<html>
<head>
<script type="text/vbscript" src="ex.vbs"></script>
</head>
<body>
</body>
</html>
```

Note: Remember to place the script exactly where you normally would write the script!

VBScript Variables
Variables are "containers" for storing information.

Do You Remember Algebra from School?
Do you remember algebra from school? x=5, y=6, z=x+y
Do you remember that a letter (like x) could be used to hold a value (like 5), and that you could use the information above to calculate the value of z to be 11?
These letters are called **variables**, and variables can be used to hold values (x=5) or expressions (z=x+y).

VBScript Variables
As with algebra, VBScript variables are used to hold values or expressions.
A variable can have a short name, like x, or a more descriptive name, like carname.
Rules for VBScript variable names:

- Must begin with a letter
- Cannot contain a period (.)
- Cannot exceed 255 characters

In VBScript, all variables are of type *variant*, that can store different types of data.

Declaring (Creating) VBScript Variables
Creating variables in VBScript is most often referred to as "declaring" variables.
You can declare VBScript variables with the Dim, Public or the Private statement. Like this:
Dim x
Dim carname
Now you have created two variables. The name of the variables are "x" and "carname".
You can also declare variables by using its name in a script. Like this:
carname="Volvo"
Now you have also created a variable. The name of the variable is "carname". However, this method is not a good practice, because you can misspell the variable name later in your script, and that can cause strange results when your script is running.
If you misspell for example the "carname" variable to "carnime", the script will automatically create a new variable called "carnime". To prevent your script from doing this, you can use the Option Explicit statement. This statement forces you to declare all your variables with the dim, public or private statement.
Put the Option Explicit statement on the top of your script. Like this:
Option Explicit
Dim carname
carname=some value

Assigning Values to Variables
You assign a value to a variable like this:
carname="Volvo"
x=10
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 "carname" has the value of "Volvo", and the variable "x" has the value of "10".

Lifetime of Variables
How long a variable exists is its lifetime.
When you declare a variable within a procedure, the variable can only be accessed within that procedure. When the procedure exits, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different procedures, because each is recognized only by the procedure in which it is declared.

If you declare a variable outside a procedure, all the procedures on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

VBScript Procedures

VBScript has two kinds procedures:

4. Sub procedure
5. Function procedure

VBScript Sub Procedures

A Sub procedure:

- is a series of statements, enclosed by the Sub and End Sub statements
- can perform actions, but **does not return** a value
- can take arguments
- without arguments, it must include an empty set of parentheses ()

```
Sub mysub()
  some statements
End Sub
```

or

```
Sub mysub(argument1,argument2)
  some statements
End Sub
```

Example (IE Only)

```
Sub mysub()
  alert("Hello World")
End Sub
```

VBScript Function Procedures

A Function procedure:

- is a series of statements, enclosed by the Function and End Function statements
- can perform actions and **can return** a value
- can take arguments that are passed to it by a calling procedure
- without arguments, must include an empty set of parentheses ()
- returns a value by assigning a value to its name

```
Function myfunction()
  some statements
  myfunction=some value
End Function
```

or

```
Function myfunction(argument1,argument2)
  some statements
  myfunction=some value
End Function
```

Example (IE Only)

```
function myfunction()
  myfunction=Date()
end function
```

How to Call a Procedure

There are different ways to call a procedure. You can call it from within another procedure, on an event, or call it within a script.

Example (IE Only)

Call a procedure when the user clicks on a button:

```
<body>
<button onclick="myfunction()">Click me</button>
</body>
```

Procedures can be used to get a variable value:

carname=findname()

Here you call a Function called "findname", the Function returns a value that will be stored in the variable "carname".

Function procedures can calculate the sum of two arguments:

Example (IE Only)

```
Function myfunction(a,b)
myfunction=a+b
End Function

document.write(myfunction(5,9))
```

The function "myfunction" will return the sum of argument "a" and argument "b". In this case 14.

When you call a procedure you can use the Call statement, like this:

Call MyProc(argument)

Or, you can omit the Call statement, like this:

MyProc argument

PHP

PHP Introduction

PHP is a server-side scripting language.

What You Should Already Know

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

- HTML/XHTML
- JavaScript

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP + MySQL

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support

PHP Installation

What do you Need?

If your server supports PHP you don't need to do anything.

Just create some .php files in your web directory, and the server will parse them for you. Because it is free, most web hosts offer PHP support.

However, if your server does not support PHP, you must install PHP.

PHP Syntax

The PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax

A PHP script always starts with **<?php** and ends with **?>**. A PHP script can be placed anywhere in the document.

On servers with shorthand-support, you can start a PHP script with <? and end with ?>.
For maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

```
<?php
?>
```

A PHP file must have a .php extension.
A PHP file normally contains HTML tags, and some PHP scripting code.
Below, we have an example of a simple PHP script that sends the text "Hello World" back to the browser:

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.
There are two basic statements to output text with PHP: **echo** and **print**.
In the example above we have used the echo statement to output the text "Hello World".

Comments in PHP
In PHP, we use **//** to make a one-line comment or **/*** and ***/** to make a comment block:

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

PHP Variables
Variables are "containers" for storing information.

Do You Remember Algebra From School?
Do you remember algebra from school? x=5, y=6, z=x+y
Do you remember that a letter (like x) could be used to hold a value (like 5), and that you could use the information above to calculate the value of z to be 11?
These letters are called **variables**, and variables can be used to hold values (x=5) or expressions (z=x+y).

PHP Variables
As with algebra, PHP variables are used to hold values or expressions.
A variable can have a short name, like x, or a more descriptive name, like carName.

Rules for PHP variable names:

- Variables in PHP starts with a $ sign, followed by the name of the variable
- The variable name must begin with a letter or the underscore character
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- A variable name should not contain spaces
- Variable names are case sensitive (y and Y are two different variables)

Creating (Declaring) PHP Variables

PHP has no command for declaring a variable.
A variable is created the moment you first assign a value to it:
$myCar="Volvo";
After the execution of the statement above, the variable **myCar** will hold the value **Volvo**.
Tip: If you want to create a variable without assigning it a value, then you assign it the value of *null*.
Let's create a variable containing a string, and a variable containing a number:

```
<?php
$txt="Hello World!";
$x=16;
?>
```

Note: When you assign a text value to a variable, put quotes around the value.

PHP is a Loosely Typed Language

In PHP, a variable does not need to be declared before adding a value to it.
In the example above, notice that we did not have to tell PHP which data type the variable is.
PHP automatically converts the variable to the correct data type, depending on its value.
In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

PHP Variable Scope

The scope of a variable is the portion of the script in which the variable can be referenced.
PHP has four different variable scopes:

- local
- global
- static
- parameter

Local Scope

A variable declared **within** a PHP function is local and can only be accessed within that function. (the variable has local scope):

```
<?php
$a = 5; // global scope

function myTest()
{
echo $a; // local scope
}

myTest();
?>
```

The script above will not produce any output because the echo statement refers to the local scope variable $a, which has not been assigned a value within this scope.
You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

Local variables are deleted as soon as the function is completed.

Global Scope
Global scope refers to any variable that is defined outside of any function.
Global variables can be accessed from any part of the script that is not inside a function.
To access a global variable from within a function, use the **global** keyword:

```
<?php
$a = 5;
$b = 10;

function myTest()
{
global $a, $b;
$b = $a + $b;
}

myTest();
echo $b;
?>
```

The script above will output 15.
PHP also stores all global variables in an array called $GLOBALS[*index*]. Its index is the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.
The example above can be rewritten as this:

```
<?php
$a = 5;
$b = 10;

function myTest()
{
$GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}

myTest();
echo $b;
?>
```

Static Scope
When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.
To do this, use the **static** keyword when you first declare the variable:

```
static $rememberMe;
```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.
Note: The variable is still local to the function.

Parameters
A parameter is a local variable whose value is passed to the function by the calling code.
Parameters are declared in a parameter list as part of the function declaration:

```
function myTest($para1,$para2,...)
{
// function code
}
```

Parameters are also called arguments. We will discuss them in more detail when we talk about functions.

PHP String Variables

A string variable is used to store and manipulate text.

String Variables in PHP

String variables are used for values that contain characters.
In this chapter we are going to look at the most common functions and operators used to manipulate strings in PHP.
After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.
Below, the PHP script assigns the text "Hello World" to a string variable called $txt:

```
<?php
$txt="Hello World";
echo $txt;
?>
```

The output of the code above will be:
Hello World
Now, lets try to use some different functions and operators to manipulate the string.

The Concatenation Operator

There is only one string operator in PHP.
The concatenation operator (.) is used to put two string values together.
To concatenate two string variables together, use the concatenation operator:

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

The output of the code above will be:
Hello World! What a nice day!
If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

The strlen() function

The strlen() function is used to return the length of a string.
Let's find the length of a string:

```
<?php
echo strlen("Hello world!");
?>
```

The output of the code above will be:
12
The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string).

The strpos() function

The strpos() function is used to search for a character/text within a string.
If a match is found, this function will return the character position of the first match. If no match is found, it will return FALSE.
Let's see if we can find the string "world" in our string:

```
<?php
echo strpos("Hello world!","world");
?>
```

The output of the code above will be:
6
The position of the string "world" in the example above is 6. The reason that it is 6 (and not 7), is that the first character position in the string is 0, and not 1.

PHP Operators
The assignment operator = is used to assign values to variables in PHP.
The arithmetic operator + is used to add values together.

Arithmetic Operators
The table below lists the arithmetic operators in PHP:

Operator	Name	Description	Example	Result
x + y	Addition	Sum of x and y	2 + 2	4
x - y	Subtraction	Difference of x and y	5 - 2	3
x * y	Multiplication	Product of x and y	5 * 2	10
x / y	Division	Quotient of x and y	15 / 5	3
x % y	Modulus	Remainder of x divided by y	5 % 2 10 % 8 10 % 2	1 2 0
- x	Negation	Opposite of x	- 2	
a . b	Concatenation	Concatenate two strings	"Hi" . "Ha"	HiHa

Assignment Operators
The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the expression on the right. That is, the value of "$x = 5" is 5.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x – y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus
a .= b	a = a . b	Concatenate two strings

Incrementing/Decrementing Operators

Operator	Name	Description
++ x	Pre-increment	Increments x by one, then returns x
x ++	Post-increment	Returns x, then increments x by one
-- x	Pre-decrement	Decrements x by one, then returns x
x --	Post-decrement	Returns x, then decrements x by one

PHP If...Else Statements
Conditional statements 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 PHP we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

---

The if Statement
Use the if statement to execute some code only if a specified condition is true.
Syntax
if (*condition*) *code to be executed if condition is true;*
The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

Notice that there is no ..else.. in this syntax. The code is executed **only if the specified condition is true**.

The if...else Statement
Use the if....else statement to execute some code if a condition is true and another code if a condition is false.
Syntax

```
if (condition)
  code to be executed if condition is true;
else
  code to be executed if condition is false;
```

Example
The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html>
<body>
```

```
<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>

</body>
</html>
```

---

*The **if...elseif....else** Statement*

Use the if....elseif...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition)
  code to be executed if condition is true;
elseif (condition)
  code to be executed if condition is true;
else
  code to be executed if condition is false;
```

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

ASP

ASP Introduction

An ASP file can contain text, HTML tags and scripts. Scripts in an ASP file are executed on the server.

What you should already know

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

- HTML / XHTML
- A scripting language like JavaScript or VBScript

What is ASP?

- ASP stands for **A**ctive **S**erver **P**ages
- ASP is a Microsoft Technology
- ASP is a program that runs inside **IIS**
- IIS stands for **I**nternet **I**nformation **S**ervices
- 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**

ASP Compatibility

- To run IIS you must have Windows NT 4.0 or later
- To run PWS you must have Windows 95 or later
- ChiliASP is a technology that runs ASP without Windows OS
- InstantASP is another technology that runs ASP without Windows

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 ASP Differ from HTML?

- 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. The ASP engine reads the ASP file, line by line, and executes the scripts in the file. Finally, the ASP file is returned to the browser as plain HTML

What can ASP do for you?

- Dynamically edit, change, or add any content of a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- The advantages of using ASP instead of CGI and Perl, are those of simplicity and speed
- Provide security - since ASP code cannot be viewed from the browser
- Clever ASP programming can minimize the network traffic

Note: Because ASP scripts are executed on the server, the browser that displays the ASP file does not need to support scripting at al

Run ASP on Your Own PC

You can run ASP on your own PC.

Your Windows PC as a Web Server

- Your own PC can act as a web server if you install IIS or PWS

- IIS or PWS turns your computer into a web server
- Microsoft IIS and PWS are free web server components

IIS - Internet Information Server
IIS is a set of Internet-based services for servers created by Microsoft for use with Microsoft Windows.
IIS comes with Windows 2000, XP, Vista, and Windows 7. It is also available for Windows NT.
IIS is easy to install and ideal for developing and testing web applications.

PWS - Personal Web Server
PWS is for older Windows system like Windows 95, 98, and NT.
PWS is easy to install and can be used for developing and testing web applications including ASP.
We don't recommend running PWS for anything else than training. It is outdated and has security issues.

Windows Web Server Versions
- Windows 7 (all editions) come with IIS 7.5
- Windows Vista Business, Enterprise and Ultimate come with IIS 7
- Windows Vista Home Premium comes with IIS 7
- Windows Vista Home Edition does not support PWS or IIS
- Windows XP Professional comes with IIS 5.1
- Windows XP Home Edition does not support IIS or PWS
- Windows 2000 Professional comes with IIS 5.0
- Windows NT Professional comes with IIS 3 and also supports IIS 4
- Windows NT Workstation supports PWS and IIS 3
- Windows ME does not support PWS or IIS
- Windows 98 comes with PWS
- Windows 95 supports PWS

ASP Basic Syntax Rules

Write Output to a Browser
An ASP file normally contains HTML tags, just like an HTML file. However, an ASP file can also contain server scripts, surrounded by the delimiters <% and %>.
Server scripts are executed on the server, and can contain any expressions, statements, procedures, or operators valid for the scripting language you prefer to use.
The response.write Command
The response.write command is used to write output to a browser. The following example sends the text "Hello World" to the browser:
Example

```
<html>
<body>
<%
response.write("Hello World!")
%>
</body>
</html>
```

There is also a shorthand method for the response.write command. The following example also sends the text "Hello World" to the browser:
Example

```
<html>
<body>
<%
```

```
="Hello World!"
%>
</body>
</html>
```

Using VBScript in ASP

You can use several scripting languages in ASP. However, the default scripting language is VBScript:

```
<html>
<body>
<%
response.write("Hello World!")
%>
</body>
</html>
```

The example above writes "Hello World!" into the body of the document.

Using JavaScript in ASP

To set JavaScript as the default scripting language for a particular page you must insert a language specification at the top of the page:

```
<%@ language="javascript"%>
<html>
<body>
<%
Response.Write("Hello World!")
%>
</body>
</html>
```

Note: JavaScript is case sensitive! You will have to write your ASP code with uppercase letters and lowercase letters when the language requires it.

Other Scripting Languages

ASP is shipped with VBScript and JScript (Microsoft's implementation of JavaScript). If you want to script in another language, like PERL, REXX, or Python, you will have to install script engines for them.

ASP Variables

A variable is used to store information.

Lifetime of Variables

A variable declared outside a procedure can be accessed and changed by any script in the ASP file.

A variable declared inside a procedure is created and destroyed every time the procedure is executed. No scripts outside the procedure can access or change the variable.

To declare variables accessible to more than one ASP file, declare them as session variables or application variables.

Session Variables

Session variables are used to store information about ONE single user, and are available to all pages in one application. Typically information stored in session variables are name, id, and preferences.

Application Variables
Application variables are also available to all pages in one application. Application variables are used to store information about ALL users in one specific application.

ASP Procedures
In ASP you can call a JavaScript procedure from a VBScript and vice versa.

Procedures
The ASP source code can contain procedures and functions:
Example

```
<html>
<head>
<%
sub vbproc(num1,num2)
response.write(num1*num2)
end sub
%>
</head>
<body>

<p>Result: <%call vbproc(3,4)%></p>

</body>
</html>
```

Insert the <%@ language="*language*" %> line above the <html> tag to write the procedure/function in another scripting language:
Example

```
<%@ language="javascript" %>
<html>
<head>
<%
function jsproc(num1,num2)
{
Response.Write(num1*num2)
}
%>
</head>
<body>

<p>Result: <%jsproc(3,4)%></p>

</body>
</html>
```

Differences Between VBScript and JavaScript
When calling a VBScript or a JavaScript procedure from an ASP file written in VBScript, you can use the "call" keyword followed by the procedure name. If a procedure requires parameters, the parameter list must be enclosed in parentheses when using the "call" keyword. If you omit the "call" keyword, the parameter list must not be enclosed in parentheses. If the procedure has no parameters, the parentheses are optional.
When calling a JavaScript or a VBScript procedure from an ASP file written in JavaScript, always use parentheses after the procedure name.

ASP Forms and User Input

The Request.QueryString and Request.Form commands are used to retrieve user input from forms.

User Input

The Request object can be used to retrieve user information from forms.

Example HTML form

```
<form method="get" action="simpleform.asp">
First Name: <input type="text" name="fname" /><br />
Last Name: <input type="text" name="lname" /><br /><br />
<input type="submit" value="Submit" />
</form>
```

User input can be retrieved with the Request.QueryString or Request.Form command.

Request.QueryString

The Request.QueryString command is used to collect values in a form with method="get". Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.

If a user typed "Bill" and "Gates" in the HTML form above, the URL sent to the server would look like this:

http://www.w3schools.com/simpleform.asp?fname=Bill&lname=Gates

Assume that "simpleform.asp" contains the following ASP script:

```
<body>
Welcome
<%
response.write(request.querystring("fname"))
response.write(" " & request.querystring("lname"))
%>
</body>
```

The browser will display the following in the body of the document:

Welcome Bill Gates

Request.Form

The Request.Form command is used to collect values in a form with method="post". Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

If a user typed "Bill" and "Gates" in the HTML form above, the URL sent to the server would look like this:

http://www.w3schools.com/simpleform.asp

Assume that "simpleform.asp" contains the following ASP script:

```
<body>
Welcome
<%
response.write(request.form("fname"))
response.write(" " & request.form("lname"))
%>
</body>
```

The browser will display the following in the body of the document:

Welcome Bill Gates
