

HTML: QuickStart Guide

Creating an Effective Website

© Copyright 2015 - All rights reserved.

In no way is it legal to reproduce, duplicate, or transmit any part of this document in either electronic means or in printed format. Recording of this publication is strictly prohibited and any storage of this document is not allowed unless with written permission from the publisher. All rights reserved.

The information provided herein is stated to be truthful and consistent, in that any liability, in terms of inattention or otherwise, by any usage or abuse of any policies, processes, or directions contained within is the solitary and utter responsibility of the recipient reader. Under no circumstances will any legal responsibility or blame be held against the publisher for any reparation, damages, or monetary loss due to the information herein, either directly or indirectly.

Respective authors own all copyrights not held by the publisher.

Legal Notice:

This book is copyright protected. This is only for personal use. You cannot amend, distribute, sell, use, quote or paraphrase any part or the content within this book without the consent of the author or copyright owner. Legal action will be pursued if this is breached.

Disclaimer Notice:

Please note the information contained within this document is for educational and entertainment purposes only. Every attempt has been made to provide accurate, up to date and reliable complete information. No warranties of any kind are expressed or implied. Readers acknowledge that the author is not engaging in the rendering of legal, financial, medical or professional advice.

By reading this document, the reader agrees that under no circumstances are we responsible for any losses, direct or indirect, which are incurred as a result of the use of information contained within this document, including, but not limited to, —errors, omissions, or inaccuracies.

Introduction

Chapter 1: Basic Tags in HTML

Chapter 2: HTML Elements

Chapter 3: Formatting in HTML

Chapter 4: HTML Phrase Tags

Chapter 5: Meta Tags

Chapter 6: Comments

Chapter 7: HTML Images

Chapter 8: HTML Lists

Chapter 9: HTML Colors

Conclusion

Free Bonus!!!

We would like to offer you our [FREE Guide](#) to jump start you on a path to improve your life & Exclusive access to our Breakthrough Book Club!!! It's a place where we offer a NEW FREE E-book every week! Also our members are actively discussing, reviewing, and sharing their thoughts on the Book of The Week and on topics to help each other Breakthrough Life's Obstacles! With a Chance to win a \$25 Gift Card EVERY Month! Please Enjoy Your [FREE Guide](#) & Access to the [Breakthrough Book Club](#)



Introduction

HTML or Hyper Text Markup Language is the most commonly used language for developing web pages. It was created in the year 1991 by Berners-Lee. The standard HTML specification, HTML 2.0, was published in the year 1995. Another major version of this language, HTML 4.01 was released in late 1999. This was the most widely used version of HTML and it is now replaced by HTML 5, released in the year 2012. This is basically an extension to its previous version HTML 4.01.

This is for?

This tutorial on HTML is designed specifically for aspiring developers and web designers. This tutorial is explained in enough detail with practical examples and a simple overview so that it can be easily understood by beginners and provides enough knowledge to design their own webpages. You can get a higher level of expertise with some practice.

Prerequisites

The prerequisites for starting this tutorial are that it should have some basic knowledge of working with Linux operating system or Windows. In addition, you should also be familiar with:

- Using a text editor like editplus, notepad, notepad++ or any other text editor.
- Creating files and directories on your computer.
- Navigation within different directories.
- Different image formats like PNG, JPG, etc.
- Typing content into a file and saving them.

Chapter 1: Basic Tags in HTML

Heading Tags

Heading is the first thing in a document and every document starts with one. In HTML, you can set your heading in different sizes. There are six sizes of headings and they can be set using the elements **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>**, and **<h6>**. The browser will add a line before and after the heading while displaying. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Heading</title>
</head>
<body>
<h1>This will be heading 1</h1>
<h2>This will be heading 2</h2>
<h3>This will be heading 3</h3>
<h4>This will be heading 4</h4>
<h5>This will be heading 5</h5>
<h6>This will be heading 6</h6>
</body>
</html>
```

This code will produce the following output:

This will be heading 1

This will be heading 2

This will be heading 3

This will be heading 4

This will be heading 5

This will be Heading 6

Paragraph Tag

You can structure your content into paragraphs using the **<p>** tag. For making your text into a paragraph, you should place it in between the opening tag **<p>** and closing tag **</p>**. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Paragraph</title>
</head>
<body>
<p>This text will be the first paragraph.</p>
```

```
<p>This text will be the second paragraph.</p>
<p>This text will be the third paragraph.</p>
</body>
</html>
```

This code will produce the following output:

This text will be the first paragraph.

This text will be the second paragraph.

This text will be the third paragraph.

Line Break Tag

If you wish to add a line break in between your text, you can use the **
** element and any text following it will start in a new line. Some tags do not need closing tags and such tags are called empty elements. The **
** element is one such empty element. They are called so because no content goes in between them.

There is a space between **br** and **/** characters and if it is not added, older browsers will fail to render the line break. If the **/** is not added and just use **
**, it won't work in XHTML, as it is invalid. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Line Break</title>
</head>
<body>
<p>Hey<br />
How are you doing today?.<br />
Good Luck<br />
Peter</p>
</body>
</html>
```

This code will produce the following output:

Hey

How are you doing today?

Good Luck

Peter

Centering Content

Sometimes you'll need to place your content at the center of the table cell or page. For such cases, you can use the tag **<center>**. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Example for Centering the Content</title>
</head>
<body>
<p>This text has the default alignment.</p>
<center>
<p>This text is centered.</p>
</center>
</body>
</html>
```

This code will produce the following the output:

This text has the default alignment.

This text is centered.

Horizontal Lines

You can visually break up the sections present in a document using the Horizontal lines. The `<hr>` tag is used for creating a horizontal line from the left margin to the right margin. This will break the line accordingly. For instance, if you want to insert a line break in between two paragraphs, you can use the code given below.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Horizontal Line</title>
</head>
<body>
<p>This is the first paragraph. This is on the top of the line break.</p>
<hr />
<p>This is the second paragraph. This is at the bottom of the line break. </p>
</body>
</html>
```

The above code will produce the following output:

This is the first paragraph. This is on the top of the line break.

This is the second paragraph. This is at the bottom of the line break.

The `<hr />` tag is also categorized as an **empty** element. This doesn't need opening tags and closing tags as there is no content going in between.

There is a space in between the characters **hr** and **/** in the `<hr />` element. If the space is omitted, older versions of browsers will have problem with the rendering the horizontal line and if the **/** is missed, it won't be a valid element in XHTML.

Preserve Formatting

In some cases your text should follow the same format as it is entered in the HTML document. You can make use of the `<pre>` tag in such cases. It means preformatted text.

Any text entered into between the tags `<pre>` and `</pre>` will have the same text format as seen in the source document. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Preserve Formatting</title>
</head>
<body>
<pre>
int main(void) {
    printf("Hello World!\n");
    // return 0;
}
</pre>
</body>
</html>
```

The above code will produce the following output.

```
int main(void) {
    printf("Hello World!\n");
    // return 0;
}
```

Nonbreaking Spaces

In some cases you do not want your browser to split the words across lines. In such cases you can use the nonbreaking space entity ** ** instead of using the normal space. Have a look at the following example for a better understanding.

Example

Let us use the words James Bond 007 for our example and we do not want the browser to split these words. We will add to the nonbreaking space entity ** ** instead of space.

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Nonbreaking Spaces</title>
</head>
<body>
<p>
```

The agent in the movie is "James Bond 007."</p>

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


Chapter 2: HTML Elements

An element in his family will be defined by starting tag. If other content can be contained within the element, it will have a closing tag. For the closing tag, the name of the element will be preceded with a /. Some of such elements are given below.

Start Tag	Content	End Tag
<i>	This is italic font content	</i>
<marquee>	This is marquee text content.	</marquee>
<p>	This is paragraph content.	</p>

Here, <i> and </i> is an element, <marquee> and </marquee> is another element. There are a few HTML elements that doesn't need closing tags, such as
, <hr />, , etc. The HTML elements which doesn't require closing tags are called as **void elements**.

The HTML document has a tree for these elements and this tree specifies on how the documents of HTML should be built. It also specifies the type of content that is to be placed in what part of the document.

HTML Tag vs. Element

An element of HTML will be defined using a starting tag. If other content is placed inside it, it will have a closing tag.

For instance, <i> is the starting tag for italic text and </i> is the closing tag for it. But, <i>This text is italic</i> is a italic text element.

Nested Elements in HTML

HTML allows you to nest HTML elements within other HTML elements. Here is an example:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Nested Elements</title>
</head>
<body>
<h2>This is <b>bold</b> heading</h2>
<p>This is <i>italic</i> text</p>
</body>
</html>
```

The above code will produce the following output

This is **bold** heading

This is *italic* text

Attributes

Till now, we have seen some of the tags like paragraph tag <p>, heading tags <h1>, <h2> and a few other tags in the most simple form. These tags can be given some extra information known as attributes.

The characteristics of an element in HTML can be defined using an attribute. This will be placed inside the opening tag of the element and they contain two parts, **name** and **value**.

- The **name** of the attribute is nothing but the property that you wish to set. For example, the font element carries attributes like color, size, type, etc., which are used for indicating the font color, font size and font type that is to be displayed on the page.
- The **value** of the attribute is nothing but the value for the property that you wish to set. The value of an attribute should always be placed inside quotations. The examples below will show you the values that can be given to the above given **name**: Red, 3 and Georgia, as font color, size and type.

You should be careful when using names and values for attributes, as they are case sensitive. However, the W3C (World Wide Web Consortium), recommends the use of lowercase attributes and attribute values in their recommendation for HTML 4.

Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Align Attribute Example</title>
</head>
<body>
<p align="right">This content will be aligned to the right</p>
<p align="center">This content will be aligned to the center</p>
<p align="left">This content will be aligned to the left</p>
</body>
</html>
```

The above code will give us the following output:

This content will be aligned to the right.

This content will be aligned to the center.

This content will be aligned to the left.

Internationalization Attributes

In HTML, there are a total of three internationalization attributes. These are available for most of the XHTML elements, but not for all.

- dir
- lang
- xml:lang

The dir Attribute

You can specify the text flow direction to the browser by using the **dir** attribute. This attribute can take two values, as shown in the table given below.

Value	Meaning
ltr	Left to right (this will be the default value)
rtl	Right to left (for languages such as Hebrew or Arabic that are read right to left)

Example

```
<!DOCTYPE html>
<html dir="rtl">
<head>
<title>Display Directions</title>
</head>
<body>
This is how IE 5 renders right-to-left directed text.
</body>
</html>
```

The above code will produce the following output:

This is how IE 5 renders right-to-left directed text.

If the **dir** attribute is placed inside the `<html>` tag, it will determine how the text of the entire document will be presented. If this attribute is used inside another tag, it will just control the direction of the text contained in that specific tag.

The lang Attribute

You can indicate the main language that is to be used in your document by using the **lang** attribute. The only reason for keeping this attribute in HTML is for the backwards compatibility with previous versions of HTML. In the new XHTML documents, this attribute is replaced with the **xml:lang** attribute.

The *lang* attributes have the values as ISO 639 standard 2-character language codes. You can check the complete list of language codes with the [HTML Language Codes: ISO 639](#).

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>English Language Page</title>
</head>
<body>
This page is using English Language
</body>
</html>
```

The xml:lang Attribute

In XHTML, the *xml:lang* attribute is used as a replacement for the original *lang* attribute used in earlier versions of HTML. As mentioned above, the *xml:lang* attribute value should be an ISO-639 country code.

Generic Attributes

There are a few other attributes that can be readily used with most of the HTML tags. The list of those attributes is given in the table below.

Attribute	Options	Function
align	right, left, center	Horizontally aligns tags
valign	top, middle, bottom	Vertically aligns tags within an HTML element.
bgcolor	numeric, hexadecimal, RGB values	Places a background color behind an element
background	URL	Places a background image behind an element
id	User Defined	Names an element for use with Cascading Style Sheets.
class	User Defined	Classifies an element for use with Cascading Style Sheets.
width	Numeric Value	Specifies the width of tables, images, or table cells.
height	Numeric Value	Specifies the height of tables, images, or table cells.
title	User Defined	"Pop-up" title of the elements.

Chapter 3: Formatting in HTML

If you are familiar with using the word processor, you probably must also be similar with making text bold, italicized or underlined. There are a total of 10 options available in HTML and XHTML, which will indicate your text will appear.

Bold Text

Any content that is placed in between `...` element will be displayed as bold text. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Bold Text</title>
</head>
<body>
<p>The following word will be in a <b>bold</b> typeface.</p>
</body>
</html>
```

The above code will produce the following output:

The following word will be in a **bold** typeface.

Italic Text

Any content that is placed in between `<i>...</i>` element will be displayed as italicized text. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Italic Text</title>
</head>
<body>
<p>The following word will be in a <i>italicized</i> typeface.</p>
</body>
</html>
```

The above code will produce the following output:

The following word will be in an *italicized* typeface.

Underlined Text

Any content that is placed in between `<u>...</u>` element will be displayed as underlined text. Here is an example.

Example

```
<!DOCTYPE html>
```



```
<html>
<head>
<title>Example for Underlined Text</title>
</head>
<body>
<p>The following word will be in an <u>underlined</u> typeface.</p>
</body>
</html>
```

The above code will produce the following output:

The following word will be in an underlined typeface.

Strike Text

Any content that is placed in between **<strike>...</strike>** element will be displayed as struck text, is a thin line through the text. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Strike Text Example</title>
</head>
<body>
<p>The following word uses a <strike>strikethrough</strike> typeface.</p>
</body>
</html>
```

The above code will produce the following output:

The following word uses a ~~strikethrough~~ typeface.

Monospaced Font

Any content placed in between the **<tt>...</tt>** element will be displayed as monospaced font. Most of the available fonts are variable width fonts. This is because different letters have different widths. For instance the letter **w** is wider than **i**. With a monospaced font, every letter will have the same width. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Monospaced Font</title>
</head>
<body>
<p>The following word uses a <tt>monospaced</tt> typeface.</p>
</body>
</html>
```

The above code will produce the following output:

The following word uses a monospaced typeface .

Superscript Text

You can use the `^{...}` element for adding a superscript on your web page. A superscript is nothing but the text with the same font with the same font size as the text surrounding it, but it will be displayed half a character's height above the surrounding text. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Superscript Text</title>
</head>
<body>
<p>The following word uses a <sup>superscript</sup> typeface.</p>
</body>
</html>
```

The above code will produce the following output:

The following word uses a ^{superscript} typeface.

Subscript Text

You can use the `_{...}` element for adding a subscript on your web page. A subscript is nothing but the text with the same font with the same font size as the text surrounding it, but it will be displayed half a character's height below the surrounding text. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Subscript Text</title>
</head>
<body>
<p>The following word uses a <sub>subscript</sub> typeface.</p>
</body>
</html>
```

The above code will produce the following output:

The following word uses a _{subscript} typeface.

Inserted Text

Any text that is placed in between the `<ins>...</ins>` element will be shown as inserted text. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Example for Inserted Text</title>
</head>
<body>
<p>I want to drink <del>cola</del> <ins>wine</ins></p>
</body>
</html>
```

The above code will produce the following output:

I want to drink cola wine

Deleted Text

Any text that is placed in between the **...** element will be shown as deleted text. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Deleted Text</title>
</head>
<body>
<p>I want to drink <del>cola</del> <ins>wine</ins></p>
</body>
</html>
```

The above code will produce the following output:

I want to drink wine

Larger Text

Any content placed inside the element **<big>...</big>** will be displayed as a one size larger font than the rest of the content surrounding it. Here is an example explaining larger text.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Larger Text</title>
</head>
<body>
<p>You can add<big>larger</big> text in middle of a sentence using this.</p>
</body>
</html>
```

This above code will produce the following output:

You can add larger text in middle of a sentence using this.

Smaller Text

Any content placed inside the element `<small>...</small>` will be displayed as a one size smaller font than the rest of the content surrounding it. Here is an example explaining smaller text.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Smaller Text</title>
</head>
<body>
<p> You can add <small>smaller</small> text in middle of a sentence using this.</p>
</body>
</html>
```

This above code will produce the following output:

You can add smaller text in middle of a sentence using this.

Grouping Content

You can use the elements `<div>` and `` for grouping several elements together for creating sections or subsections in a page. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Div Tag</title>
</head>
<body>
<div id="menu" align="middle" >
<a href="/index.htm">HOME</a> |
<a href="/about/contact_us.htm">CONTACT</a> |
<a href="/about/index.htm">ABOUT</a>
</div>

<div id="content" align="left" bgcolor="white">
<h4>Some Content</h4>
<p>The content will go here.....</p>
</div>
</body>
</html>
```

This will produce the following output:

[HOME](#) | [CONTACT](#) | [ABOUT](#)

SOME CONTENT

The content will go here.....

On the other hand, the `` element can only be used for grouping inline elements. In cases where you need to group a part of a paragraph or a sentence together, you can make use of the `` element. Here

is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Span Tag</title>
</head>
<body>
<p>This is the example of <span style="color:green">span tag</span> and the <span style="color:red">div
tag</span> along with CSS</p>
</body>
</html>
```

The above code will produce the following output.

This is the example of span tag and the div tag along with CSS

The above tags are used alongside CSS, which allow you to attach a style to a particular section of a page.

Chapter 4: HTML Phrase Tags

The phrase tags in HTML are designed for performing specific purposes. They look similar to other tags like ``, `<i>`, `<pre>`, and `<tt>`, which we have already seen in our previous chapters. In this chapter we will go through the important phrase tags in HTML.

Emphasized Text

Any content that is placed in between the `...` element will be shown as emphasized text. Here is an example.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Emphasized Text</title>
</head>
<body>
<p>The following word uses a <em>emphasized</em> typeface.</p>
</body>
</html>
```

The above code will produce the following output:

The following word uses an *emphasized* typeface.

Marked Text

Any content that is placed in between the `<mark>...</mark>` element will be shown as marked text, with a yellow link. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Marked Text Example</title>
</head>
<body>
<p>The following word has been <mark>marked</mark> with yellow</p>
</body>
</html>
```

The above code will produce the following output:

The following word has been **marked** with yellow.

Strong Text

Any content that is placed in between the `...` element will be shown as important text. Here is an example.

Example

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>Strong Text Example</title>
</head>
<body>
<p>The following word uses a <strong>strong</strong> typeface.</p>
</body>
</html>
```

The above code will produce the following output:

The following word uses a **strong** typeface.

Text Abbreviation

In HTML, you are allowed to abbreviate your text. You can do it by placing the text inside the opening **<abbr>** and closing **</abbr>** tags. The full description should be added in the title attribute and nothing else, if present. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Text Abbreviation</title>
</head>
<body>
<p>My best friend's name is <abbr title="Abigail">Abi</abbr>.</p>
</body>
</html>
```

The above code will produce the following output:

My best friend's name is Abi.

Acronym Element

You can indicate your text as an acronym in HTML by using the **<acronym>** element. You should place your text in between the opening tag **<acronym>** and closing tag **</acronym>** for indicating it as an acronym. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Acronym Example</title>
</head>
<body>
<p>This chapter covers marking up text in <acronym>XHTML</acronym>.</p>
</body>
</html>
```

The above code will produce the following output:

This chapter covers marking up text in XHTML.

Text Direction

The element `<bdo>...</bdo>` stands for Bi-Directional Override. You can override the text direction using this element. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Text Direction</title>
</head>
<body>
<p>This text goes from left to right.</p>
<p><bdo dir="rtl">This text goes from right to left.</bdo></p>
</body>
</html>
```

The above code will produce the following output:

This text goes from left to right.

This text goes from right to left.

Special Terms

You can introduce special terms on your web page on for doing it; you can use the element `<dfn>...</dfn>`. This is very much similar to italicized text in the middle of a paragraph.

You would typically use the element `<dfn>` when you introduce a key term for the first time. The recent browsers will render the text of the `<dfn>` element in and italicized format. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Special Terms Example</title>
</head>
<body>
<p>The following word is a <dfn>special</dfn> term.</p>
</body>
</html>
```

The above code will produce the following output:

The following word is a special term.

Quoting Text

In HTML, you can quote a passage from a different source. For doing it, you should place the content in between the opening tag `<blockquote>` and closing tag `</blockquote>`.

Any text placed inside this element will usually be intended from left and right edges of the text surrounding it. Sometimes it uses italicized font. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Blockquote Example</title>
</head>
<body>
<p>The following description of XHTML is taken from the W3C Web site:</p>

<blockquote>XHTML 1.0 is the W3C's first Recommendation for XHTML, following on from earlier work on
HTML 4.01, HTML 4.0, HTML 3.2 and HTML 2.0.</blockquote>
</body>
</html>
```

The above code will produce the following output:

The following description of XHTML is taken from the W3C Web site:

XHTML 1.0 is the W3C's first Recommendation for XHTML, following on from earlier work on HTML 4.01, HTML 4.0, HTML 3.2 and HTML 2.0.

Short Quotations

You can add double quotes inside a sentence by using the `<q>...</q>` element. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Double Quote</title>
</head>
<body>
<p>Paul is in Spain, <q>I think I am right</q>.</p>
</body>
</html>
```

The above code will produce the following output:

Paul is in Spain, I think I am right.

Text Citations

When you are quoting some text, by placing the source between the opening and closing tags `<cite></cite>`, you can indicate the source. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Citations</title>
</head>
<body>
<p>This HTML tutorial is derived from <cite>W3 Standard for HTML</cite>.</p>
```

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

The above code will produce the following output:

This HTML tutorial is derived from W3 Standard for HTML.

Computer Code

If you are adding any programming code to your web page, you should place it inside the opening tag **<code>** and closing tag **</code>**. The content placed inside this element will usually be presented in the monospaced font, like any other called in other programming books. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example Computer Code</title>
</head>
<body>
<p>This is regular text. <code>This is code.</code> This is regular text.</p>
</body>
</html>
```

The above code will produce the following output:

This is regular text. This is code . This is regular text.

Keyboard Text

In some situations, you will need your reader to type text using his keyboard. In such situations you can make use of the element **<kbd>...</kbd>**. This element will indicate a text that is to be typed. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Keyboard Text</title>
</head>
<body>
<p>Regular text. <kbd>This is inside kbd element</kbd> Regular text.</p>
</body>
</html>
```

The above code will produce the following output:

Regular text. This is inside kbd element Regular text.

Programming Variables

Usually, this element will be used along with the elements **<code>** and **<pre>** for indicating that the content placed inside this element is variable. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example Variable Text</title>
</head>
<body>
<p><code>document.write("<var>user-id</var>")</code></p>
</body>
</html>
```

The above code will produce the following output:

```
document.write("user-id")
```

Program Output

The sample output of a program can be indicated using the element **<samp>...</samp>**. Script can also be indicated using this. However, it is usually used for documenting coding concepts or programming. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example Program Output</title>
</head>
<body>
<p>The result of the program is <samp>Hello World!</samp></p>
</body>
</html>
```

The above code will produce the following output:

The result of the program is Hello World!

Address Text

You can give the address using the **<address>...</address>** element. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Address</title>
</head>
<body>
<address>1719 c, David Drive, Marietta, Atlanta, Georgia </address>
</body>
</html>
```

The above code will produce the following output:

1719 c, David Drive, Marietta, Atlanta, Georgia

Chapter 5: Meta Tags

In HTML, you are allowed to specifying metadata, which is the additional important information regarding the document, in different ways. You can use the META elements for including name/value pairs for describing the properties of the document, such as document author, author, expiry date, the list of keywords, etc.,

Such additional information can be provided using the **<meta>** tag. There are no closing tags for the Meta tag and hence it is an empty element. Though it is an empty element, but the information can be passed through its attributes.

In general, you can add more than one meta tags to your document, basing on the information that you are adding to your document. The physical appearance of your document will not have any effect because of meta tags. From the application point of view, it doesn't matter if meta tags are included or not.

Adding Meta Tags to Your Documents

You can provide a Meta data to your pages by adding `<meta>` tags within The document's header, represented by the tags `<head>` and `</head>`. In addition to their core attributes, they can have the following attributes:

Attribute	Description
Name	Name for the property. Can be anything. Examples include, keywords, description, author, revised, generator etc.
content	Specifies the property's value.
scheme	Specifies a scheme to interpret the property's value (as declared in the content attribute).
http-equiv	Used for http response message headers. For example http-equiv can be used to refresh the page or to set a cookie. Values include content-type, expires, refresh and set-cookie.

Specifying Keywords

Important keywords that are related to the document can be specified using `<meta>` tag. These keywords will be later used by various search engines. They will also be used for indexing your page for search. Here is an example.

Example

In the below example, we will add the words Ducati, Aprilia and BMW as the important keywords related to the document.

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="Ducati, Aprilia, BMW" />
</head>
<body>
```

```
<p>Hello World!</p>
</body>
</html>
```

The above code will produce the following output:

Hello World!

Document Description

You can provide a short description of your document by using `<meta>` tags. Similar to keywords, the subscription will also be used by search engines while indexing the web page for searching purposes. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Meta Tags</title>
<meta name="keywords" content="Ducati, Aprilia, BMW" />
<meta name="description" content="Learning about Meta Tags." />
</head>
<body>
<p>Hello World!</p>
</body>
</html>
```

Document Revision Date

The `<meta>` tag can be used for providing information regarding the last updation of the document. Various web browsers use this information while refreshing your web page. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="Ducati, Aprilia, BMW" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="revised" content="Topgear, 7/7/15" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

Document Refreshing

You can use the `<meta>` tag for specifying the duration on when to auto refresh your page. Here is an example.

Example

If you wish to make your web page refreshed automatically for every 10 seconds, use the below syntax.

Syntax:

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="Ducati, Aprilia, BMW" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="revised" content="Topgear, 7/7/15" />
<meta http-equiv="refresh" content="10" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

Page Redirection

Using <meta> tag, you can redirect your web page to other web pages. You can actually specify the duration when you want to redirect the page by specifying the number of seconds. Here is an example.

Example

In the following example, the current page will be redirected to another page after 10 seconds. If you leave the content attribute unspecified, the current web page will be redirected immediately.

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="Ducati, Aprilia, BMW" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="revised" content="Topgear, 7/7/15" />
<meta http-equiv="refresh" content="5; url=http://www.topgear.com" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

Setting Cookies

Cookies are nothing but a data that is stored on your computer in the form of small text files. Cookies are exchanged between the web server and the web browser for keeping track on the information that the web application needs.

Using <meta> tag, the cookies can be stored on the client side and of this information can be later used by the web server for tracking the visitor of its site.

Example

In the following example, the current page will be redirected to another page after 10 seconds. If you leave the content attribute unspecified, the current web page will be redirected immediately.

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="Ducati, Aprilia, BMW" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="revised" content="Topgear, 7/Jul/15" />
<meta http-equiv="cookie" content="userid=xyz; expires=Wednesday, 08-Aug-15 23:59:59 GMT;" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

If the expiration date and time are not included, the cookie will then be considered as a session cooking. Session cookies will be deleted when the browser is closed.

Setting Author Name

You can set an author name in a web page using meta tag. See an example below:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="Ducati, Aprilia, BMW" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="author" content="Mahnaz Mohtashim" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

Specify Character Set

You can use <meta> tag to specify character set used within the webpage.

Example

By default, Web servers and Web browsers use ISO-8859-1 (Latin1) encoding to process Web pages. Following is an example to set UTF-8 encoding:

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="Ducati, Aprilia, BMW" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="author" content="Mahnaz Mohtashim" />
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
```

```
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

To serve the static page with traditional Chinese characters, the webpage must contain a <meta> tag to set Big5 encoding:

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="Ducati, Aprilia, BMW" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="author" content="Mahnaz Mohtashim" />
<meta http-equiv="Content-Type" content="text/html; charset=Big5" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```


Chapter 6: Comments

Comments are nothing but pieces of code that are ignored by web browser. Adding comments to the HTML code is a very good practice, especially for complex documents. Comments indicate different sections of a document. They can also be used for indicating other notes to other persons looking at the code. By increasing the code readability, comments help users to understand code easily.

For adding a comment in HTML, you need to place the content in between the tags `<!-- ... -->`. Any content that is placed inside these tags will be considered as comments and the browser will completely ignore them. Here is an example

Example

```
<!DOCTYPE html>
<html>
<head> <!-- Document Header Starts -->
<title>This is document title</title>
</head> <!-- Document Header Ends -->
<body>
<p>The content of the document goes here.....</p>
</body>
</html>
```

This code will produce the output which displays the content placed inside the paragraph tag but it will hide the content placed inside the comment tags.

The content of the document goes here.....

Valid vs. Invalid Comments

Comments cannot be nested; this means that you cannot place a comment inside another comment. You should always ensure that there are no spaces given at the start of the comment string. The '--' (double dash sequence) may not appear within a comments except in the closing tag (`-->`). Here is an example.

Example

In this example, we will give a valid comment and it will be ignored by the browser.

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Comment</title>
</head>
<body>
<!-- This comment is valid -->
<p>The content of the document goes here.....</p>
</body>
</html>
```

The above code will produce the following output:

The content of the document goes here.....

The following example will have an invalid comment. Browsers do not hide invalid comments and they display them as content. We will add an extra space in between the opening `<` and `!`.

```
<!DOCTYPE html>
<html>
<head>
<title>Example for Invalid Comment</title>
</head>
<body>
<!-- This comment is not a valid one -->
<p>The content of the document goes here.....</p>
</body>
</html>
```

The above code will produce the following output:

```
<!-- This comment is not a valid one -->
```

The content of the document goes here.....

Multiline Comments

So far, we have only discussed about single line comments. However, HTML also supports the multi-line comments.

You can increase the code readability by adding comments in multiple lines by using a special beginning and ending tags `<!--` and `-->` that are placed in front of the first line and after the last line as given in the example shown below.

Example

```
<!DOCTYPE html><html>
<head>
<title>Multiline Comments</title>
</head>
<body>
<!--
This is an example of multiline comment and
it can be extended to any number of
lines you like.
-->
<p>The content of the document goes here.....</p>
</body>
</html>
```

The above code will produce the following output:

The content of the document goes here.....

Conditional Comments

The conditional comments do not work on any other browser other than Internet Explorer. Other browsers ignore conditional comments. The support for conditional comments started from Internet Explorer 5. They can be used for giving conditional instructions for different Internet Explorer versions. Here is an example.

Example

```
<!DOCTYPE html><html>
<head>
<title>Example for Conditional Comments</title>

<!--[if IE 6]>
  Special instructions for IE 6 here
<![endif]-->

</head>
<body>
<p>The content of the document goes here.....</p>
</body>
</html>
```

Conditional comments can be helpful in situations where you will need to use different style sheets, depending on the version of IE.

Using Comment Tag

A part of the HTML code can be made as a comment and there are a few browsers supporting the `<comment>` tag. Here is an example.

Example

```
<!DOCTYPE html><html>
<head>
<title>Using Comment Tag</title>
</head>
<body>
<p>This is <comment>not</comment> Internet Explorer.</p>
</body>
</html>
```

The above code will produce the following output if you're using IE:

This is Internet Explorer.

The above code will produce the following output if you're using a different browser other than IE:

This is not Internet Explorer.

Commenting Script Code

In cases where you are using VBScript or JavaScript with your HTML code, it is wise to place the script code inside proper comments. This will make sure that they can be used with older browsers. Here is an example.

Example

```
<!DOCTYPE html><html>
<head>
<title>Commenting Script Code</title>
<script>
<!--
  document.write("Hakuna Matata!")
```

```
//-->
</script>
</head>
<body>
<p>Hakuna, Matata!</p>
</body>
</html>
```

The above code will produce the following output:

Hakuna Matata!

Hakuna, Matata!

Commenting Style Sheets

In cases where you are using style sheets with your HTML code, it is wise to place the style sheet code inside proper comments. This will make sure that they can be used with older browsers. We will learn about style sheets in a different chapter. Here is an example.

Example

```
<!DOCTYPE html><html>
<head>
<title>Commenting Style Sheets</title>
<style>
<!--
.example {
border:1px solid #4a7d49;
}
//-->
</style>
</head>
<body>
<div class="example">Good, Morning!</div>
</body>
</html>
```

The above code will produce the following output:

Good, Morning!

Chapter 7: HTML Images

Images can be very helpful to beautify and to explain the complex concepts of your web page in a simple way. In this chapter, we will add images to your webpages in simple steps.

Insert Image

In HTML, you can add any image to your web page with the `` tag. The syntax for using the `` tag is given below.

Syntax:

```

```

The `` tag is an empty one, it means that there will be no closing tag for it. You can only give the list of attributes to this tag. Here is an example.

Example

For trying the below example, you need to keep the .htm file and the image within the same directory.

```
<!DOCTYPE html>
<html>
<head>
<title>Using Image in Webpage</title>
</head>
<body>
<p>Simple Image Insert</p>

</body>
</html>
```

The above code will produce an output with the image used.

Simple Image Insert

HTML supports image formats like JPEG, PNG or GIF. You can use any of these image formats basing on your comfort. You should always ensure that you specify the correct file name of the image in `src` attribute. Always remember that the image names are case sensitive.

There is a mandatory attribute called the **alt** attribute, which will specify an alternate text for the image, in cases where the image fails to be displayed.

Set Image Location

It is recommended that you keep your images in a specific directory separately. We will put our .htm file in the home directory and the images in a subdirectory called **images**, within the home directory. Here is an example.

Example

Here, we will assume the image location to be `"/html/image/test.png"`, for this example.

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Using Image in Webpage</title>
</head>
<body>
<p>Simple Image Insert</p>

</body>
</html>
```

The above code will produce an output with the specified image.

Set Image Width/Height

Depending on your requirements, you can set the height and width of an image with the help of the **height** and **width** attributes. The height and width for the image can either be specified in terms of its percentage with its actual size or in terms of pixels. Not that the original resolution of the image will change if the ratios are not maintained. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Set Image Width and Height</title>
</head>
<body>
<p>Setting image width and height</p>

</body>
</html>
```

The above code will produce an output image with the width 200 and height 150.

Set Image Border

Every image by default will have a border surrounding it. You can specify the thickness of the border using the **border** attribute, in terms of pixels. If you wish to remove the border around your image, you should set the thickness to 0. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Set Image Border</title>
</head>
<body>
<p>Setting image Border</p>

</body>
</html>
```

The above code will produce an output with the border 3.

Set Image Alignment

All the images, by default, will be aligned to the left. You can specify the alignment of the image using the **align** attribute to set the image to the right or center. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Set Image Alignment</title>
</head>
<body>
<p>Setting image Alignment</p>

</body>
</html>
```

The above code will produce an output image aligned to the right, with a border 3.

Chapter 8: HTML Lists

In HTML, information can be specified using three types of lists. A list must contain at least one list element. Lists contain:

**** - This is an unordered list. The list items will be listed using plain bullets.

**** - This is an ordered list. The list items in this list will be listed with different schemes of numbers.

<dl> - This is a definition list. The items in the definition list will be arranged in the same way as they are in a dictionary.

HTML Unordered Lists

And unordered list can be defined as a collection of items with no special sequence or order. You can create this list by using the `` tag. All the items in this list will be marked using bullets. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
<ul>
<li>Apple</li>
<li>Mango</li>
<li>Banana</li>
<li>Pear</li>
</ul>
</body>
</html>
```

The above code will produce the following output:

Apple

Mango

Banana

Pear

The type Attribute

You can actually specify the type of bullet to be used by using the **type** attribute of the `` tag. The default a bullet time is a disc. The other possible options are:

```
<ul type="disc">
```

```
<ul type="square">
```

```
<ul type="circle">
```

Example

Following is an example where we used `<ul type="square">`

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
  <ul type="square">
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
```

```
<li>Pear</li>
</ul>
</body>
</html>
```

The above code will produce the following output:

Apple

Mango

Banana

Pear

Example

Following is an example where we used `<ul type="disc">` :

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
  <ul type="disc">
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
    <li>Pear</li>
  </ul>
</body>
</html>
```

The above code will produce the following output:

Apple

Mango

Banana

Pear

Example

Following is an example where we used `<ul type="circle">` :

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
  <ul type="circle">
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
    <li>Pear</li>
  </ul>
</body>
</html>
```

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

The above code will produce the following output:

Apple

Mango

Banana

Pear

HTML Ordered Lists

You can use the order list in HTML when you are required to place the list items in a numbered list, instead of using the regular bulleted list. You can place the items in an ordered list by using the **** tag. The number in for an ordered list starts with 1 and it will be incremented by 1 with each element added to this list using ****. Here is an example.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol>
<li>Apple</li>
<li>Mango</li>
<li>Banana</li>
<li>Pear</li>
</ol>
</body>
</html>
```

The above code will produce the following output:

Apple

Mango

Banana

Pear

The type Attribute

You can specify the numbering type you like by using the **type** attribute of the **** tag. The number will be the default numbering type. The other possible options are the following:

<ol type="A"> - Upper-Case Letters.

<ol type="a"> - Lower-Case Letters.

<ol type="I"> - Upper-Case Numerals.

<ol type="i"> - Lowercase Numerals.

<ol type="1"> - Default-Case Numerals.

Example

We will use the **<ol type="1">** for this example.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
```



```
</head>
<body>
  <ol type="1">
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
    <li>Pear</li>
  </ol>
</body>
</html>
```

The above code will produce the following output:

Apple

Mango

Banana

Pear

Example

We will use the `<ol type="1">` for this example.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
  <ol type="1">
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
    <li>Pear</li>
  </ol>
</body>
</html>
```

The above code will produce the following output:

Apple

Mango

Banana

Pear

Example

We will use the `<ol type="i">` for this example.

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

```
<head>
<title>HTML Ordered List</title>
</head>
<body>
  <ol type="i">
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
    <li>Pear</li>
  </ol>
</body>
</html>
```

The above code will produce the following output:

Apple

Mango

Banana

Pear

Example

We will use the `<ol type="A">` for this example.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
  <ol type="A">
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
    <li>Pear</li>
  </ol>
</body>
</html>
```

The above code will produce the following output:

Apple

Mango

Banana

Pear

Example

We will use the `<ol type="a">` for this example.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
  <ol type="a">
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
    <li>Pear</li>
  </ol>
</body>
</html>
```

The above code will produce the following output:

Apple

Mango

Banana

Pear

The start Attribute

You can specify the starting point with a specific number to start a list with the help of the **start** attribute of the `` tag. The other possible options the following (we will start at the numbering with 3).

```
<ol type="1" start="3"> - Numerals starts with 3
```

```
<ol type="a" start="3"> - Letters starts with c
```

```
<ol type="A" start="3"> - Letters starts with C
```

```
<ol type="i" start="3"> - Numerals starts with iii
```

```
<ol type="I" start="3"> - Numerals starts with III
```

Example

Following is an example where we used `<ol type="i" start="3" >`

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
  <ol type="i" start="3">
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
    <li>Pear</li>
  </ol>
```

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

The above code will produce the following output:

Apple
Mango
Banana
Pear

HTML Definition Lists

XHTML and HTML support a list style called the **definition list**. Here are the entities will be listed like as they are in an encyclopaedia or dictionary. Using the definition list is ideal for representing a list of items, glossary, or other value/name lists.

The given tags will be used by the definition list.

`<dl>` - Defines the start of the list

`<dt>` - A term

`<dd>` - Term definition

`</dl>` - Defines the end of the list

Here is an example using `<dt>`

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Definition List</title>
</head>
<body>
<dl>
<dt><b>HTML</b></dt>
<dd>This stands for Hyper Text Markup Language</dd>
<dt><b>HTTP</b></dt>
<dd>This stands for Hyper Text Transfer Protocol</dd>
</dl>
</body>
</html>
```

The above code will produce the following output:

HTML
This stands for Hyper Text Markup Language
HTTP
This stands for Hyper Text Transfer Protocol

Chapter 9: HTML Colors

Colors play a very important role in beautifying your web page, making it good to look and feel it. On the page level, you can specify the colors by using the <body> tag. You can also use the **bgcolor** attribute for setting the colors what different tags.

The following attributes are available with the <body> tag, with which different colors can be set.

vlink - This will set the color for the links that are visited (links clicked once or more)

link - The color of the linked text can be set using this.

alink - The color for the selected links or active links can be set using this.

text - The body text color can be set with this.

bgcolor - The color of the background can be set with this.

HTML Color Coding Methods

There are three different methods in HTML for setting setting colors for the web page. They are as follows:

Color names - you can directly specify the names of the colors like red, blue or green.

Hex codes - You can specify the color using a six digit hex code. Every color can be represented by the red, blue and green colors present in them. The hex code is nothing but a representation of this.

Color decimal or percentage values - This value will be specified by using the rgb() property.

Now we will have a look at these coloring schemes in detail.

HTML Colors - Color Names

You can directly specify the name of the color for setting it as the background or text color. There are a total of 16 basic colors listed by W3C, that validate with the HTML validator. Most of the major browsers support more than 200 different colors.

W3C Standard 16 Colors

The list of 16 standard colors, recommended by W3C is given below.

	Black		Gray		Silver		White
	Yellow		Lime		Aqua		Fuchsia
	Red		Green		Blue		Purple
	Maroon		Olive		Navy		Teal

Example

Here is an example where we said to the background color using the color name.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Colors by Name</title>
</head>
<body text="blue" bgcolor="green">
<p>Use different color names for for body and table and see the result.</p>
```



```

<table bgcolor="black">
<tr>
<td>
<font color="white">This text will appear white on black background.</font>
</td>
</tr>
</table>
</body>
</html>

```

HTML Colors - Hex Codes

The hex code is nothing but the six digit the presentation for a color. The first two digits of the hex code represent the value of red, the next two digits represented the value of green and the last two digits represented the value of blue. The hex code will be RRGGBB.

You can take the hexadecimal value from graphics software like MS paint, Paintshop Pro or Adobe Photoshop.

A hash sign # or pound will precede the hexadecimal code. Given below is a table with the colors using hex code.

Color	Color HEX
	#000000
	#FF0000
	#00FF00
	#0000FF
	#FFFF00
	#00FFFF
	#FF00FF
	#COCOCO
	#FFFFFF

Example

Here are the examples to set background of an HTML tag by color code in hexadecimal:

```

<!DOCTYPE html>
<html>
<head>
<title>HTML Colors by Hex</title>
</head>
<body text="#0000FF" bgcolor="#00FF00">
<p>Use different color hex for for body and table and see the result.</p>
<table bgcolor="#000000">
<tr>
<td>

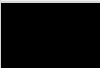








```

```
<font color="#FFFFFF">This text will appear white on black background.</font>
</td>
</tr>
</table>
</body>
</html>
```

HTML Colors - RGB Values

The color value can be specified with the **rgb()** property. This takes 3 values as the input for the colors red, green and blue, respectively. The range of each color varies from integer values 0 to 255 or as a percentage.

The list of the RGB values for few colors is given below.

Color	Color RGB
	rgb(0,0,0)
	rgb(255,0,0)
	rgb(0,255,0)
	rgb(0,0,255)
	rgb(255,255,0)
	rgb(0,255,255)
	rgb(255,0,255)
	rgb(192,192,192)
	rgb(255,255,255)

Example

Here are the examples to set background of an HTML tag by color code using rgb() values:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Colors by RGB code</title>
</head>
<body text="rgb(0,0,255)" bgcolor="rgb(0,255,0)">
<p>Use different color code for for body and table and see the result.</p>
<table bgcolor="rgb(0,0,0)">
<tr>
<td>
<font color="rgb(255,255,255)">This text will appear white on black background.</font>
</td>
</tr>
</table>
</body>
```

</html>

Note: Not all browsers support the rgb() color property. You should be careful on its usage for this reason.

Browser Safe Colors

There are a total of 216 colors that are supposed to be browser safe. They are computer independent colors. These colors range from the 000000 to FFFFFF in hex code. All computers having the 256 color palette support these.

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF

CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

Conclusion

Creating a webpage is something that has been wreathed in mysticism and cyber mumbo jumbo for years, but I hope that I've shown you that it's not as scary as you might think. In fact, HTML is actually fairly straight forward. IF you're looking for a hobby that can really accomplish something and is easy to explain, then you've come to the right place. These are the foundations upon which you can build your webpage and push on to greater things like CSS, Java, JavaScript, C++, and the list goes on and on. Remember, that you can never really get in over your head if you master and truly understand what it is that you're working on in the moment.

If something doesn't make sense to you or is confusing, my best suggestion to you is to open up your TextEdit or Notepad and start working on it. Pull it up in your browser and have a look at what it does for your page. Taking everything step by step will inevitably lead you to the path of success. There is never a reason to jump ahead without fully understanding what it is that you've done.

If you ever find yourself having a need to get back to the basics or you just don't remember how to do something simple, come back to this book and have a look at the fundamentals. With everything that I have presented to you in this book, you should be able to get the grasp on the beginnings of a webpage and if you can get the beginning, it's just a matter of taking the next step piece by piece. So good luck and get out there and make an amazing webpage.

Thank you for Reading! I Need Your Help...

Dear Reader,

I Hope you Enjoyed "**HTML: Quick Start Guide Creating an Effective Website**". I have to tell you, as an Author, I love feedback! I am always seeking ways to improve my current books and make the next ones better. It's readers like you who have the biggest impact on a book's success and development! So, tell me what you liked, what you loved, and even what you hated. I would love to hear from you, and I would like to ask you a favor, if you are so inclined, would you please share a minute to review my book. Loved it, Hated it - I'd just enjoy your feedback. As you May have gleaned from my books, reviews can be tough to come by these days and You the reader have the power make or break the success of a book. If you'd be so kind to [CLICK HERE](#) to review the book, I would greatly appreciate it! Thank you so much again for reading "**HTML: Quick Start Guide Creating an Effective Website**" and for spending time with me! I will see you in the next one!

Check Out More From The Publisher...

Social Media: Master Social Media Marketing - Facebook, Twitter, YouTube & Instagram

by Grant Kennedy

<http://www.amazon.com/Social-Media-Marketing-Facebook-Instagram-ebook/dp/B018Y68SWS>

Survival: The Survival Guide for Preppers, Make Yourself Ready Through Hunting, Fishing, Canning, and Foraging

by Jack Campbell

<http://www.amazon.com/Survival-Preppers-Permaculture-Bushcraft-Hydroponics-ebook/dp/B01573FBP8>

Krav Maga: Dominating Solutions to Real World Violence

by George Silva

<http://www.amazon.com/Krav-Maga-Dominating-Solutions-Violence-ebook/dp/B01A2BL6CW>

Day Trading: Dominate The Market - Stocks, Options, & Forex

by Warrick Liversedge

<http://www.amazon.com/Day-Trading-Dominate-Options-Covered-ebook/dp/B01A3LJCMW>

SEO: Marketing Strategies to Dominate the First Page

by Grant Kennedy

<http://www.amazon.com/SEO-Marketing-Strategies-analytics-optimization-ebook/dp/B01ACB7LQM>

JavaScript:

Quick start Guide – Effective JavaScript Programming

© Copyright 2016 - All rights reserved.

In no way is it legal to reproduce, duplicate, or transmit any part of this document in either electronic means or in printed format. Recording of this publication is strictly prohibited and any storage of this document is not allowed unless with written permission from the publisher. All rights reserved.

The information provided herein is stated to be truthful and consistent, in that any liability, in terms of inattention or otherwise, by any usage or abuse of any policies, processes, or directions contained within is the solitary and utter responsibility of the recipient reader. Under no circumstances will any legal responsibility or blame be held against the publisher for any reparation, damages, or monetary loss due to the information herein, either directly or indirectly.

Respective authors own all copyrights not held by the publisher.

Legal Notice:

This book is copyright protected. This is only for personal use. You cannot amend, distribute, sell, use, quote or paraphrase any part or the content within this book without the consent of the author or copyright owner. Legal action will be pursued if this is breached.

Disclaimer Notice:

Please note the information contained within this document is for educational and entertainment purposes only. Every attempt has been made to provide accurate, up to date and reliable complete information. No warranties of any kind are expressed or implied. Readers acknowledge that the author is not engaging in the rendering of legal, financial, medical or professional advice.

By reading this document, the reader agrees that under no circumstances are we responsible for any losses, direct or indirect, which are incurred as a result of the use of information contained within this document, including, but not limited to, —errors, omissions, or inaccuracies.

Introduction

Chapter 1: Introduction to JavaScript Programming

Chapter 2: JavaScript - Specifications and Features

Chapter 3: JavaScript – Syntax

Chapter 4: Enabling and Disabling JavaScript in your Web Browser

Chapter 5: JavaScript Data Types

Chapter 6: Operators

Chapter 7: Case Statements and Loops

Conclusion

Free Bonus!!!

We would like to offer you our [FREE Guide](#) to jump start you on a path to improve your life & Exclusive access to our Breakthrough Book Club!!! It's a place where we offer a NEW FREE E-book every week! Also our members are actively discussing, reviewing, and sharing their thoughts on the Book of The Week and on topics to help each other Breakthrough Life's Obstacles! Please Enjoy Your [FREE Guide](#) & Access to the [Breakthrough Book Club](#)



<https://publishfs.leadpages.co/the-breakthrough-book-club-d/>

Introduction

Thank you very much for choosing this book! JavaScript programming is one of the unique interpreted programming languages that is absolutely essential in producing World Wide Web content. Like most of the programming languages out there, JavaScript too is a C language based syntax that is object oriented and produces real life based outputs.

This book is going to give you an insight to what JavaScript programming is all about and the things you can create once you are done with it. You will learn about the different concepts involved in JavaScript, its syntax and how you can start programming in it. This programming language will help you greatly in webpages and its designs. There are more advanced versions of JavaScript available, JavaScript will help you cover the elementary concepts that are not only easy but are also helpful in understanding the more advanced versions of the language.

JavaScript is going to help you become better at creating algorithms, understanding basics of any object oriented programming language and also create your own content. This book is going to give you all the information you will need in order to understand and start with JavaScript Programming as we go through the chapters.

Chapter 1: Introduction to JavaScript Programming

JavaScript is a high-level, untyped, dynamic and yet interpreted programming language that has been standardized in the ECMAScript language specification. JavaScript is also one of the three main programming languages that bolster the World Wide Web content production, along with HTML (Hyper Text Markup Language) and CSS (Cascading Style Sheets). A big majority of the internet websites employ JavaScript in their makeup and it is also supported by almost all the modern web browsers with plug-ins or any other external software to make it compatible with the system or the browser. It is one of the programming languages that is prototype based and contains great world class features that makes it on par with the other multi paradigm languages and also supports object-oriented (outputs are real time objects), functional and imperative programming styles.

JavaScript also supports an Application Programming Interface or an API that supports various features such as text, dates, characters, arrays, matrices, regular expressions and so on. One down side to it is that it does not support any Input/output or an I/O which include networking, graphic facilities, or even storage and relies on the host environment for these features as they are embedded on them.

Though the infamous misconceptions still take place, Java and JavaScript are completely unrelated except for the standard and syntactic libraries and also have very different semantics. While it is true that the syntax of JavaScript is based on C language, the semantics and design is derived from and influenced by the Self and Scheme Programming languages.

JavaScript is known for its versatility in the compilation and synthesizing software required to run a program, it can also be executed in environments that are not Web based such as Notepad, PDF documents, desktop widgets and even website applications. Earlier, JavaScript was traditionally implemented just as an interpreted language on the client side but later evolved into a *just in time* compilation in the more recent versions of the browsers. The latest and the faster versions of JavaScript include Virtual Machines (VMs) and platforms are built upon them. JavaScript I, increasingly popular in the game development, server side network programming with runtime environments such as Node.js and even in the creation of the desktop and mobile applications.

History

JavaScript Beginnings in Netscape

Brendan Eich is the creator of JavaScript, who was still employed in Netscape Communications Corporation and he managed to finish developing the idea of JavaScript in about 10 days in the Year 1995 around the month of May. Netscape Communications Corporation was indeed in a race competing against Microsoft in creating the user adoption of Web Technologies and platforms. Netscape considered itself a client-server offering a portable version of Sun Microsystems' Java with a distributed Operating System and providing an environment in which applets could be run. The most distinguishing feature of Java is that it was a competitor of C and C++ which was aimed at the professional programmers whereas Netscape wanted to be more of a light weight interpreted language that would complement Java and also appeal to the nonprofessional programmers, much like Microsoft's Visual Basic.

It was initially developed under the name Mocha, but was also often referred to as LiveScript when it was first released and was shipped to the beta releases of Netscape Navigator 2.0 in September 1995, but after which it was renamed as JavaScript. This was the official name Netscape Browser version 2.0 B3 used after it was deployed.

It became clear that the change of the name from LiveScript to JavaScript happened at the time when Netscape was adding support for Java Technology in its Netscape Navigator Web Browser. This tended to cause a storm among the programmers who immediately assumed that JavaScript was some sort of a spin-off of Java programming language and it was all a part of Netscape's marketing ploy that went off to give JavaScript the cachet of what was then the new, hot website programming language much similar to what Java enjoyed for years.

Though there was a popular misconception in the 90s when JavaScript was first introduced, the programming language was highly inspired by an earlier website scripting language which was developed by Nombas known as C--, not to be confused with the actual C— which was created in 1997. Brendan Eich, the creator of JavaScript then said he had, however, never heard of C— before creating LiveScript in the year 1995. Actually Nombas had pitched their webpage embedded scripting system to Netscape which was a new concept at that time and was known by ViolaWWW. Later, Nombas actually dropped C— and started to offer JavaScript in their ScriptEase product and was part of the TC39 group that standardized ECMAScript in the 90s.

Server side JavaScript

In the year 1995, Netscape was trying implementation of the language of server side scripting with Netscape Enterprise Server in the month of December soon after the release of browser version of JavaScript. There was a resurgence of the server side interpretation of JavaScript in the mid - 2000s such as the Node.js.

Adoption by Microsoft

Microsoft Windows released many scripting technologies in the year 1996 including VBScript and Jscript and so on. Microsoft released Jscript as a reverse engineered implementation of Netscape Communications Corporation's JavaScript which was released in the year 1996 on the 16th of July as a part of Internet Explorer 3 and as well as acting as an available server in the Internet Information Server. IE3 included a lot of Microsoft's first support for CSS (Cascading Style Sheet) and the extended versions of HTML. In each case of the implementation of JavaScript, it was noticeably different to that found in Netscape Navigator at that time. These glaring differences in the languages used in the scripting of a webpage made it difficult for web designers and programmers and left them confused as to what they should do in order to make a single website work well in both browsers leading to 'the best and most viewed in Netscape' and 'Best viewed in Internet Explorer' logos that became a big part of the first of the 'Best Browser Wars'. The programmers and web designers soon overcame this roadblock all thanks to JavaScript, one of the most cutting edge languages that allowed them to cross platforms and it was

compatible with almost all standard driven web. Without JavaScript the web developer faced a hard time to write and design web pages for both the major browsers and could not afford the time. After the release of Internet Explorer 4 Microsoft started to introduce the concept of Dynamic HTML, but the different language implementation in the proprietary document object models remained as ales to widespread take up of JavaScript on the World Wide Web.

Standardization

In year 1996 during the month of November, Netscape made an announcement that they had submitted JavaScript to the Ecma International for a consideration to become an industry standard and subsequently the work done by Netscape bolstered it to become the standardized version of ECMAScript. In the following year, ECMA International led to publish its first edition of the ECMA-262 and its specifications. In the month of June of 1998, they made several changes and modifications to adapt it to the ISO/IEC-16262 standard and hence the second version was released. The subsequent third edition of ECMA-262 was released in the month of December in 1999.

However, the development of the fourth edition of the ECMAScript standard was stopped half way through and was never completed. Whereas they proceeded to make the 5th edition which was released in December 2009. The sixth and the current edition of the ECMAScript being used today was released in June 2015.

Later Developments

Suffice to say, JavaScript became one of the most popular programming languages on the Web because of its crass platform features and other specifications. However, initially many of the professional programmers and web developers denigrated the language because of its target audience and said that it was for 'amateurs' and for web authors, and so on. The emergence of Ajax on the scene returned JavaScript to the spotlight and captured the attention of more professional programmers and computer experts. In order to improve the JavaScript Programming practices, they started to proliferate the comprehensive frameworks and libraries which resulted in the increased usage of the JavaScript even outside the web browsers which can be seen in the proliferation of server side JavaScript platforms.

However in the year 2009, the CommonJS projected was founded, aimed at the goal of establishing a common standard library mainly for JavaScript program development outside the Web browser.

With reaching heights of popularity of the single page applications which were heavily embedded with JavaScript, it was soon increasingly used as a compile target for most of the *source to source* compilers from both static languages and dynamic languages. EMscripten in particular, and other optimized JIT compilers, in tandem with asm.js that was compatible with AOT companies like OdinMonkey, have paved a way for enabling C and C++ program to be compiled into JavaScript and caused the programs to be executed in near native speeds, causing JavaScript as the "Assembly Language of the Web" much like what Perl enjoys today.

Trademark

Today, 'JavaScript' is trademarked by the Oracle Corporation. It is still being used under the license for technology that was invented and executed by Netscape communications and currently holds the entitlement to establishments such as the Mozilla Foundation.

Chapter 2: JavaScript - Specifications and Features

As described above JavaScript is a lightweight, interpreted programming language, which is now being used in creating network specific applications. JavaScript is also complementary and integrated with Java which is very easy to implement when embedded along with HTML and CSS. It is an open and cross platform programming language.

Now let us explore the various specifications of JavaScript before getting to the syntax. First and foremost you need to understand the basic functionality of JavaScript before you go on to designing a dynamic web page and applications.

Client side JavaScript

One of the most popular forms of programming versions is the Client-side JavaScript, that is, the script that needs to be included in or referenced by a HTML document for the programming to be interpreted by the user.

This means that the web page need not be a static webpage written with static HTML but can include various programs and functions to interact with the user and dynamically create HTML content by controlling the browser.

This client-side mechanism exhibits many advantages over the usual CGI server-side scripts. For example, the programmer can use JavaScript to check if the user has entered the right user ID in the form field in a web page.

The JavaScript code ensures that the entries listed by the user are valid or not and then proceed to execute commands to submit the form to the web browser. It can also be used to trap user initiated events such as the link navigation, button clicks and other actions that are either implicitly or explicitly initiated by the user.

Advantages of JavaScript

The following lists the merits of using JavaScript while designing a webpage:

- Less server interaction – That is the user input can be validated before sending the page off the server which helps lessen server traffic. This means that you are offering less load on the server which prevents from congestion.
- Immediate response to the users – JavaScript does not wait for a page to reload to notify the user of anything they have forgotten or missed.
- Increase in interactivity – There are more options for you to create interfaces using which the webpage can interact with the user, like when the user hovers over a marquee with the mouse or activates it using the keyboard.
- Richer interfaces – JavaScript is used to include various items such as the *drag and drop* components and sliders to provide a better interface experience for your web users.

Limitations of using JavaScript

JavaScript cannot be treated as a full-fledged programming language due to the demerits and the features it lacks listed below:

- The most important version of JavaScript, the client-side JavaScript does not allow the reading and writing of the files due to various security purposes.
- There is no support available for JavaScript to be used in networking applications.
- JavaScript also lacks having any multiprocessor or multithreading capabilities.

JavaScript Development Tools

One of the major advantages of using JavaScript is that it does not require any expensive tools and software to run. JavaScript can even be executed using simple text editors such as Notepad. It does not even require a compiler since it is an interpreted language used inside the context of the web browser.

To make matters simple, various corporations have come forward with great editing tools designed specifically for JavaScript. Some of them are listed down below –

- Microsoft FrontPage – FrontPage is a software created by Microsoft that is used to edit HTML codes but also provides a good interface for JavaScript. It provides web developer a number of tools required by them to execute JavaScript Tools and assist in the creation of interactive websites.
- Macromedia HomeSite 5 – Similar to Microsoft FrontPage, HomeSite 5 is a popular HTML and JavaScript editor created by Macromedia and can be used to manage personal websites efficiently.

Where is JavaScript today?

As we have discussed before the ECMAScript 5 standard will be the first update released in over four years and JavaScript 2.0 conforms to Edition 5 of ECMAScript standard, where the differences between the two is extremely minor.

The exact specifications of JavaScript 2.0 can be found in the following URL : <http://www.ecmascript.org/>

Today although both Netscape's JavaScript and Microsoft's Jscript still support the features of ECMAScript standards and conform to it, they are still not a part of it.

Chapter 3: JavaScript – Syntax

Syntax of JavaScript is the written text that will allow you to give instructions to the web browsers that will follow the guideline in order to perform various operations. JavaScript statements are usually placed within the `<script> ... </script>` HTML tags in a web page. There is also an option for you to place your JavaScript code anywhere within your webpage in the `<script>` tags but it is normally recommended that you keep it within the `<head>` tags.

The `<script>` tags always alerts the browser program to read and interpret each and every line between these tags as a script. A simple JavaScript syntax will look like this:

```
<script>
```

JavaScript code

```
</script>
```

There are two important attributes of script tag –

- **Type** – This attribute is usually what is recommended to indicate the scripting language that is used in the web site and its values used usually be set to “text/javascript”
- **Language** – This attribute specifies the kind of scripting language used in the web design. Its value is usually “javascript”, even though the recent versions of HTML (and its successor XHTML) have phased out the use of this attribute.

And so your JavaScript code will look something like this:

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

JavaScript code

```
</script>
```

How to write your very own JavaScript Program?

The most famous test program for any programming language is the printing of “Hello World” in the output screen. Let us dig in further on how we can execute that. In this program we will be adding an optional HTML comment that is surrounded by our JavaScript code. Next, we have to conjure a function `document.write` which embeds a written string into the HTML document.

This function is generally used to write HTML, text or both. The following is the example program to print “Hello World”.

```
<html>
```

```
<body>
```

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

```
<!--
```

```
    document.write("Hello World!")
```

```
//-->
```

```
</script>
```

```
</body>
```

```
</html>
```


OUTPUT:

Hello World!

Whitespace and Line Breaks

One of the main things to remember while typing a JavaScript program is that it ignores all spaces, newlines, tabs, and so on. Using Tabs, spaces and newlines are completely your wish and you are allowed to format your program in which ever way it is convenient and easy for you. If you indent your programs to look neat and consistent, then it will be easy for both you and a third party who reads the code and understand it quickly. This also makes it easier for you to spot errors in the program line.

Semicolons are Optional

In C, C++ and Java it is usually required that every complete command is truncated with a semicolon, and the same rule is implemented in JavaScript for simple statements. However, in JavaScript, it allows you to omit a semicolon, only when you place your statements in different lines so that the browser can understand the command.

NOTE: However, it is healthy to practice programming with the use of semicolons since they are an integral part of so many other programming languages.

Case Sensitivity

It is important to note that JavaScript is a case sensitive language, that means the language and the case of the keywords, function names, variables and all of the other identifiers must be maintained at the same case at all times throughout the program.

For example, the identifier char and CHAR have different meanings and can be used as individual keywords in the same program.

NOTE – It is important to pay close attention to the name of your variables and the functions since they cannot match a keyword. If they do then it will throw an error.

Comments in JavaScript

JavaScript is known to support both the styles of C language and C++ in its comments, that is –

- Any text between the `'//'` symbol, and all the other characters that follow are treated as a comment by JavaScript. That is, it will not read these lines and it is used for your personal purposes only.
- Any text and characters in between in the symbols `/*` and `*/` will be treated as a comment in JavaScript. These comments can span multiple lines at a time unlike `//`.
- JavaScript recognizes the HTML comment sequence that opens with a `<!--`. This is similar to the ones used in C, the `//`.
- The HTML closing sequence `-->` is not recognized by JavaScript and so a comment should be closed with `//-->`.

Chapter 4: Enabling and Disabling JavaScript in your Web Browser

Almost all the modern browsers that we use today come with a built in support for JavaScript that can be enabled or disabled by the users manually. This chapter focuses on enabling and disabling the JavaScript settings and support system in your browsers such as the - Internet Explorer, Google Chrome, Mozilla Firefox and Opera to name a few.

JavaScript in Internet Explorer:

The following instructions will guide you to turn on and turn off the JavaScript support in your Internet Explorer –

- Go to *Tools* → Select *Internet Options* from the menu
- Click on the *Security* Tab from the dialog box
- Select the *Custom* Level button.
- Scroll down the drop box until you find *Scripting* Options in the list.
- After locating the *Active Scripting*, select the *Enable* radio button.
- Finally click on the *okay* button and exit the window.

In order to disable the JavaScript support in your Internet Explorer settings, all you need to do is follow the same set of instructions expect selecting *Disable* radio button under *Active Scripting*.

JavaScript in Mozilla Firefox

The following instructions will guide you to turn on and turn off the JavaScript support in your Mozilla Firefox –

- Open a Mozilla Firefox tab and type *about: config* in the address bar of the page.
- After which a warning dialog will open and you need to select '*I'll be careful, I promise!*'
- You will find a list of all the *configure options* in the browser in a drop box.
- Type *javascript.enabled* in the search bar of the web browser.
- Here you will find an option to enable and disable JavaScript by right clicking on the option → then *select toggle*

JavaScript in Google Chrome

The following instructions will guide you to turn on and turn off the JavaScript support in your Google Chrome –

- Select the Chrome menu which is located at the top right hand corner of your web browser.
- Click on the *settings* option
- Select *Show Advanced Settings* which is located at the end of the page.
- Click on the content settings button which is right below the *Privacy* Section.
- You will need to select "*Do not allow any site to run JavaScript*" in order to disable JavaScript settings or select "*Allow all sites to run JavaScript (recommended)*"

JavaScript in Opera

The following instructions will guide you to turn on and turn off the JavaScript support in your Opera browser -

- Open the *Tools* menu and select *Preferences*.
- Click on the *Advanced* option from the dialog box.
- Select *Content* from the drop down box.
- Click on the *Enable JavaScript* checkbox
- Finally click on the *okay* button and exit the menu.

To disable the JavaScript support in your Opera web browser, you should leave the *Enable JavaScript* button unchecked.

Warning for Browsers that do not support JavaScript

If there is anything you need to use JavaScript for in the browsers that don't support JavaScript, then you are able to display a warning message using the `<noscript>` tags.

The noscript tag is added immediately after the script block as written below:

```
<html>
```

```
<body>
```

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

```
document.write("Hello World!!")
```

```
</script>
```

```
<noscript>
```

```
Sorry JavaScript is required in order to proceed.
```

```
</noscript>
```

```
</body>
```

```
</html>
```

In case the user's browser does not support JavaScript or is not enabled then the message will be displayed on the screen from `</noscript>`. JavaScript enjoys flexibility to include JavaScript in a HTML file as follows -

- Program code inside `<head> ... </head>` section.
- Program code inside `<body> ... </body>` section.
- Program code inside `<head> ... </head>` and `<body> ... </body>` section.
- Program code in an external file and then include the `<head> ... </head>` section.

Chapter 5: JavaScript Data Types

JavaScript Data types

One of the most basic and fundamental building blocks of a programming language is the set of data types it supports. There are various types and collections of data types that are used and here are the types of values that is allowed to be used and manipulated in JavaScript.

JavaScript consists of the following basic data types –

- **Strings and characters:** It is a special character or a collection of characters. For example, 'This is a text string'
- **Numbers:** They are numerical values which can be expressed normally, with decimals (floating points), binary and hexadecimals, for example, 145, 23.98 etc.
- **Boolean:** They are logical expressions, example AND logic, OR logic, etc.

In JavaScript the most rudimentary type of data is null and undefined, where each of which can define only one value. Apart from these primitive data types, JavaScript also supports a composite data type, that is a collection of basic data types, wrapped into a single object.

NOTE: It is important to note that JavaScript does not distinguish between integer values and floating point values. All numbers are taken and represented in the form of floating point numbers using the 64 bit floating point format which is defined in the IEEE 754 standard.

JavaScript Variables

One of the main attributes of any programming language are the variables. JavaScript, like many offers variables similar to C, C++ and Java. Variables are usually a named container. Their use is similar to that of an algebraic variable, where the letter or a word contains different numerical and alphanumeric values at some point in the program.

In JavaScript, the basic keyword to create a variable is *var* and you must declare it in the beginning of the program before you start using it in your program.

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

```
<!--
```

```
var amount;
```

```
var rate;
```

```
//-->
```

```
</script>
```

There is also an option of declaring multiple variables with a same var keyword at the same time. This is known as dynamic initialization of a variable.

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

```
<!--
```

```
var amount, rate;
```

```
-->
```

```
</script>
```

This type of declaring the variable name and type is known as variable initialization. You can initialize a variable at any point of time and anywhere in the program as and when you need a variable. All variables must be used at some point or else JavaScript throws an error for redundant variables.

Here is the type of initializations that can be done throughout the program. You can also change the values of the variables at any time inside the programs. Here is an example of variable initialization –

```
<Script type = "text/javascript">
<!--
var name = "Rose";
var age;
age = 21;
//-->
</script>
```

NOTE – Once a *var* keyword declares or initializes a variable in the program, the same variable cannot be initialized in the document again.

Like it was mentioned before, JavaScript is an untyped language. This means that the variable declared in a JavaScript can be of any data type, a number, character, string or even a Boolean expression. This feature isn't possible in so many of the other programming languages. Most of them need to specify the type of variable being declared in the program as and when they type it. JavaScript interprets the type of change that the variables goes through the run time of the program and adapts the data type accordingly.

JavaScript Variable Scope

In any programming language you need to understand the scope of a variable. The scope is the area or the region of the program in which the variable is valid. JavaScript supports only two types of scope –

- Global variable – A global variable is declared in the main body of the program or even before the main program and its scope is valid from the line it is declared and is valid throughout the program in your JavaScript code.
- Local Variables – A local variable is valid only inside a section of the program, specifically inside a function or parenthesis. Any reference to that variable outside the scope will cause JavaScript to throw errors.

It is important to note that, inside the body of the function, the global variable is given less precedence than a local variable of the same name. If you happen to declare a variable or a function name same as a global variable it is recommended to, effectively hide it. Take a look at this example –

```
<html>
<body onload = checkscope();>
<script type = "text/javascript">
<!--
var myVar = "global"; // Declare a global variable
function checkscope( ) {
var myVar = "local"; // Declare a local variable
document.write(myVar);
}
//-->
</script>
```

</body>

</html>

This is the OUTPUT of the following program

Local

JavaScript Variable Names

There are so many factors that you should remember before you name your variable in a JavaScript program. Here is a list of things you need to consider before creating a variable –

- It is the rule of thumb that you absolutely refrain from using any of the already reserved keywords of JavaScript as your variable names. The list of keywords that are reserved is mentioned in the next section. For example, words like *Boolean* or *break* are not valid variable names.
- In JavaScript, the variable names may contain a number but it cannot start with a numeral (from 0 to 9). They must either begin with a letter or an underscore. For example, a valid variable name is *test123* or *_123test* but *123test* is invalid.
- Like mentioned before JavaScript is case sensitive and it is possible for the same word to be a variable with different characterization. For example, *money* and *Money* are two different words and can be used as two different variables.

JavaScript Keywords or Reserved Words

Keywords are an essential fundamental indicators of a programming language and they may be used for various purposes. These words are reserved and cannot be used to name any of the JavaScript variables, characters, functions, methods, objects or loop tables.

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

Chapter 6: Operators

Operators are the expressions that is used to manipulate a set of variables. For instance, you need to perform a mathematical operation $3 + 6$ is equal to 9. The variables that are used to manipulate, i.e. 3 and 6 are known as operands and the '+' sign is known as the operator. Like many other programming languages, JavaScript supports the following types of operators.

Arithmetic Operators

Comparison Operator

Logical Operators

Assignment Operators

Conditional Operators

Now let us discuss them in further detail.

Arithmetic Operators

Arithmetic Operators are used to perform various mathematical operations inside the JavaScript program. The following are used in JavaScript.

Let us assume that the operands a and b hold the values 20 and 30.

Sr.No	Operator and Description
1	Addition – '+' This operator gives the added values of two operands Ex: $A + B = 30$
2	Subtraction – '-' This operator gives the subtracted values of two operands Ex: $A - B = -10$
3	Multiplication – '*' This operator gives the multiplied value of two operands Ex: $A * B = 600$
4	Division – '/' This operator gives the quotient after dividing two operands Ex: $B / A = \frac{3}{4}$
5	Modulus – '%' This operator gives the remainder value after dividing two operands Ex: $B \% A = 10$

6	<p>Increment – ‘++’</p> <p>This operator adds one to the operand.</p> <p>Ex: A++ will give 21</p>
7	<p>Decrement – ‘--’</p> <p>This operator subtracts the value of the operand by one.</p> <p>Ex: A-- will give 19</p>

NOTE: The addition operator is used to add both numerals and strings. For example, “a” + 20 will give “a20”.

Example program for Arithmetic Operators:

The following program illustrates how to use and manipulate variables using arithmetic operators.

```
<html>
```

```
  <body>
```

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

```
      <!--
```

```
        var a = 33;
```

```
        var b = 10;
```

```
        var c = "Test";
```

```
        var linebreak = "<br />";
```

```
        document.write("a + b = ");
```

```
        result = a + b;
```

```
        document.write(result);
```

```
        document.write(linebreak);
```

```
        document.write("a - b = ");
```

```
        result = a - b;
```

```
        document.write(result);
```

```
        document.write(linebreak);
```

```
        document.write("a / b = ");
```

```
        result = a / b;
```

```
document.write(result);  
document.write(linebreak);
```

```
document.write("a % b = ");  
result = a % b;  
document.write(result);  
document.write(linebreak);
```

```
document.write("a + b + c = ");  
result = a + b + c;  
document.write(result);  
document.write(linebreak);
```

```
a = ++a;  
document.write("++a = ");  
result = ++a;  
document.write(result);
```

```
document.write(linebreak);
```

```
b = --b;  
document.write("--b = ");  
result = --b;  
document.write(result);  
document.write(linebreak);
```

```
//-->
```

```
</script>
```

```
</body>
```

```
</html>
```

OUTPUT of the following program is

`a+b=43`
`a-b=23`
`a/b=3.3`
`a%b=3`
`a+b+c=43`
`++a=35`
`--b=8`

You can change the values of the variables in your program and try it.

Comparison Operator

Comparison Operators are used in comparing the values of two operands and then making a conclusion from it. JavaScript supports the following comparison operators –

Assume that the variables A and B hold the values 20 and 30 respectively -

Sr.No	Operator and Description
1	Equal – ‘==’ This operator checks if the values of two operands are equal, and if they are equal then the output is true. Ex: (A == B) is not true.
2	Not Equal – ‘!=’ This operator checks if the values of two operands are not equal, and if they are not equal then the output is true. Ex: (A != B) is true.
3	Greater than – ‘>’ This operator checks if the value of first operand is greater than the other, and if it is greater, then the output is true. Ex: (A > B) is not true.
4	Less than – ‘<’ This operator checks if the value of second operand is greater than the other, and if it is greater, then the output is true. Ex: (A < B) is true.
5	Greater than or Equal to – ‘>=’ This operator checks if the value of first operand is greater than or equal to the other, and if it is greater or equal to the next operand, then the output is true. Ex: (A >= B) is not true.
6	Less than or Equal to – ‘<=’

This operator checks if the value of first operand is lesser or equal to the second operand, and if it is lesser than or equal, then the output is true.

Ex: (A <= B) is true.

EXAMPLE PROGRAM:

The following JavaScript program will help you understand how the logical operators are initialized and how it works in a program.

```
<html>
<body>

<script type="text/javascript">
  <!--
    var a = 10;
    var b = 20;
    var linebreak = "<br />";

    document.write("(a == b) => ");
    result = (a == b);
    document.write(result);
    document.write(linebreak);

    document.write("(a < b) => ");
    result = (a < b);
    document.write(result);
    document.write(linebreak);

    document.write("(a > b) => ");
    result = (a > b);
    document.write(result);
    document.write(linebreak);

    document.write("(a != b) => ");
    result = (a != b);
    document.write(result);
    document.write(linebreak);

    document.write("(a >= b) => ");
    result = (a >= b);
    document.write(result);
    document.write(linebreak);

    document.write("(a <= b) => ");
    result = (a <= b);
    document.write(result);
```

```

        document.write(linebreak);
    //-->
</script>
</body>
</html>

```

OUTPUT of the following program is

```

(a == b) => false
(a < b) => true
(a > b) => false
(a != b) => true
(a >= b) => false
(a <= b) => true

```

Logical Operators

Logical Operators are the ones used in executing Boolean expressions in the JavaScript program. Here are the logical operators supported by JavaScript –

Assume that the variables A and B hold the values 20 and 30 -

Sr.No	Operator and Description
1	<p>Logical AND – ‘&&’</p> <p>This operator checks if both the operators are non-zero, and if so the condition is true.</p> <p>Ex: (A && B) is true.</p>
2	<p>Logical OR – ‘ ’</p> <p>This operator checks if any one of the operators are non-zero, and if so the condition is true.</p> <p>Ex: (A B) is true.</p>
3	<p>Logical NOT – ‘!=’</p> <p>This operator is used to reverse the logical state of an operand. When this condition becomes true then the Logical NOT will make it false.</p> <p>Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false.</p> <p>Ex: !(A && B) is false.</p>

Example program

The following problem will aid in understanding how the logical operators are used and what outputs you will be yielding –

```

<html>
  <body>

```

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

```
<!--
```

```
var a = true;
```

```
var b = false;
```

```
var linebreak = "<br />";
```

```
document.write("(a && b) => ");
```

```
result = (a && b);
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
document.write("(a || b) => ");
```

```
result = (a || b);
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
document.write("! (a && b) => ");
```

```
result = !(a && b);
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
//-->
```

```
</script>
```

```
</body>
```

```
</html>
```

OUTPUT

```
(a && b) => false
```

```
(a || b) => true
```

```
!(a && b) => true
```

Bitwise Operators

Bitwise operators are operators that can manipulate only one at a time. Boolean operators are also a part of these.

Assume that the variables A and B are 2 and 3 respectively -

Sr.No	Operator and Description
1	Bitwise AND – ‘&’ This operator performs the Boolean AND operation on each bit of its integer arguments. Ex: (A & B) is 2.
2	BitWise OR – ‘ ’ This operator performs the Boolean OR operation on each bit of its integer arguments. Ex: (A B) is 3.

3	<p>Bitwise XOR – ‘^’</p> <p>This operator performs the Boolean exclusive OR Operation on each bit of its integer arguments and the condition becomes true when both the operands are either zero or non-zero.</p> <p>Ex: (A ^ B) is 1.</p>
4	<p>Bitwise Not – ‘~’</p> <p>This is a unary operator and it reverses the operand before operation.</p> <p>Ex: (~B) is -4.</p>
5	<p>Left Shift – ‘<<’</p> <p>According to the number specified, this operator moves the bits to the left the number of places in the second operand. The new bits are filled with zeros and shifting a value left by one position in binary system, is equivalent of multiplying it by 2, and shifting it by 2 places is like multiplying it by 4 and so on.</p> <p>Ex: (A << 1) is 4.</p>
6	<p>Right Shift – ‘>>’</p> <p>This operator is known as the Binary Right Shift Operator and it moves the left operand’s value to the right by the number of bits specified by the right operand.</p> <p>Ex: (A >> 1) is 1.</p>
7	<p>Right shift with Zero – ‘>>>’</p> <p>This operator is similar to the binary right shift operator, except that the bits shifted in on the left are replaced with a zero.</p> <p>Ex: (A >>> 1) is 1.</p>

Example Program

```

<html>
  <body>

  <script type="text/javascript">
    <!--
      var a = 2; // Bit presentation 10
      var b = 3; // Bit presentation 11
      var linebreak = "<br />";

      document.write("(a & b) => ");
      result = (a & b);
      document.write(result);
    -->
  </script>
  </body>
</html>

```

```
document.write(linebreak);
```

```
document.write("(a | b) => ");  
result = (a | b);  
document.write(result);  
document.write(linebreak);
```

```
document.write("(a ^ b) => ");  
result = (a ^ b);  
document.write(result);  
document.write(linebreak);
```

```
document.write("(~b) => ");  
result = (~b);  
document.write(result);  
document.write(linebreak);
```

```
document.write("(a << b) => ");  
result = (a << b);  
document.write(result);  
document.write(linebreak);
```

```
document.write("(a >> b) => ");  
result = (a >> b);  
document.write(result);  
document.write(linebreak);
```

```
//-->
```

```
</script>
```

```
</body>
```

```
</html>
```

OUTPUT

(a & b) => 2

(a | b) => 3

(a ^ b) => 1

(~b) => -4

(a << b) => 16

(a >> b) => 0

Assignment Operators

Assignment Operators helps you assign values to your variable in a JavaScript Code.

Sr.No	Operator and Description
1	Simple Assignment - '=' This operator is used to assign the value of the first operator to the next.

Ex: $C = A + B$, the added values of the variables are stored in c

2 Add and Assignment – ‘+=’

This operator is used to simultaneously add itself and assign it to the variable in the left.

Ex: $B += A$ is same as $B = B + A$

3 Subtract and Assignment – ‘-=’

This operator is used to simultaneously subtract itself and assign it to the variable in the left.

Ex: $B -= A$ is equivalent to $B = B - A$

4 Multiply and Assignment – ‘*=’

This operator is used to simultaneously multiply itself and assign it to the variable in the left.

Ex: $B *= A$ is equivalent to $B = B * A$

5 Divide and Assignment – ‘/=’

This operator is used to simultaneously divide itself and assign the quotient to the variable in the left.

Ex: $B /= A$ is equivalent to $B = B / A$

6 Modules and Assignment – ‘%=’

This operator is used to simultaneously divide itself and assign the remainder to the variable in the left.

Ex: $B \% A$ is equivalent to $B = B \% A$

NOTE: The same logic can be applied to all the Bitwise Operators and they are $\ll=$, $\gg=$, $\&=$, $|=$ and $\^=$.

EXAMPLE

```
<html>
```

```
<body>
```

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

```
<!--
```

```
var a = 33;
```

```
var b = 10;
```

```
var linebreak = "<br />";
```

```
document.write("Value of a => (a = b) => ");
```

```
result = (a = b);
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
document.write("Value of a => (a += b) => ");
```

```
result = (a += b);
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
document.write("Value of a => (a -= b) => ");
```

```
result = (a -= b);
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
document.write("Value of a => (a *= b) => ");
```

```
result = (a *= b);
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
document.write("Value of a => (a /= b) => ");
```

```
result = (a /= b);
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
document.write("Value of a => (a %= b) => ");
```

```
result = (a %= b);
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
//-->
```

```
</script>
```

```
</body>
```

```
</html>
```

OUTPUT

Value of a => (a = b) => 10
Value of a => (a += b) => 20
Value of a => (a -= b) => 10
*Value of a => (a *= b) => 100*
Value of a => (a /= b) => 10
Value of a => (a %= b) => 0

Miscellaneous Operators

The two important operators in JavaScript are the conditional Operator and the Typeof operator.

Conditional Operator (? ;)

The conditional operator is like an 'if else' statement, that is, it first evaluates an expression using the other operators listed above and then determines if it is true or false and then executes the given statement depending on the condition assigned.

EXAMPLE PROGRAM

```
<html>
<body>

<script type="text/javascript">
  <!--
    var a = 10;
    var b = 20;
    var linebreak = "<br />";

    document.write ("((a > b) ? 100 : 200) => ");
    result = (a > b) ? 100 : 200;
    document.write(result);
    document.write(linebreak);

    document.write ("((a < b) ? 100 : 200) => ");
    result = (a < b) ? 100 : 200;
    document.write(result);
    document.write(linebreak);
  //-->
</script>
</body>
</html>
```

OUTPUT:

((a > b) ? 100 : 200) => 200
((a < b) ? 100 : 200) => 100

Typeof Operator

This is a type of unary operator that should be placed before a single operand which can be of any type and its value indicating the data type of the operand.

The typeof operator uses "number", "string" or "Boolean" only if the operand is number, string or Boolean

respectively and returns true or false accordingly.

This is a list of typeof return values that this operator generates -

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"

EXAMPLE PROGRAM:

```
<html>
```

```
<body>
```

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

```
<!--
```

```
var a = 10;
```

```
var b = "String";
```

```
var linebreak = "<br />";
```

```
result = (typeof b == "string" ? "B is String" : "B is Numeric");
```

```
document.write("Result => ");
```

```
document.write(result);
```

```
document.write(linebreak);
```

```
result = (typeof a == "string" ? "A is String" : "A is Numeric");
```

```
document.write("Result => ");
```

```
document.write(result);
```

```
    document.write(linebreak);  
  //-->  
</script>  
</body>  
</html>
```

OUTPUT:

```
Result => B is String  
Result => A is Numeric
```


Chapter 7: Case Statements and Loops

While executing various algorithms in JavaScript, there will be times where you will have to repeat a statement or an operation more than once or adopt certain set of paths. In these cases, you need *case statements* that will help you check a certain condition and help you execute the necessary follow up statements.

JavaScript supports conditional statements that are used to check a certain condition before executing the relevant action. One of the main case statements used is the If ... Else statement.

If-Else Statement

The If-Else Statement is very similar to the conditional Operator that was discussed in operators. There are three forms of if-else statements :

If statement

If ... Else statement

If ... Else if ... Statement

If Statement

The if statement is a singular, fundamental control line that allows JavaScript to make decisions based on the condition and execute the necessary actions.

Syntax

If (expression)

{

Statements to be executed if the expression is true

}

First, JavaScript reads the if condition in brackets. Then the values are substituted and checked if the statement is true. In case it is true then it will follow the instructions written in the parenthesis below the condition. If it is false these statements are ignored entirely.

EXAMPLE PROGRAM:

```
<html>
```

```
<body>
```

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

```
<!--
```

```
var age = 20;
```

```
if( age > 18 ){
```

```
document.write("<b>Qualifies for driving</b>");
```

```
}
```

```
//-->
```

```
</script>
```

```
<p>Set the variable to different value and then try...</p>
```

```
</body>
```

```
</html>
```

OUTPUT:

Qualifies for driving.

If ... Else Statement

This is the next version of the if statement that allows you to perform the if statement based upon the conditions provided but then executes a set of instructions for the false case also.

Syntax

```
If ( expression )
```

```
{
```

```
Statements that need to be executed if the condition is true.
```

```
}
```

```
Else
```

```
{
```

```
Statements that need to be executed if the condition is false.
```

```
}
```

Like the if statement, the condition is evaluated. If the expression is true then the statements in the parenthesis are executed. If the expression is false the statements in the parenthesis below the else statement.

EXAMPLE PROGRAM:

```
< html>
```

```
<body>
```

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

```
<!--
```

```
var age = 15;
```

```
if( age > 18 ){
```

```
document.write("<b>Qualifies for driving</b>");
```

```
}
```

```
else{
```

```
document.write("<b>Does not qualify for driving</b>");  
}  
//-->  
</script>
```

<p>Set the variable to different value and then try...</p>

</body>

</html>

OUTPUT

Does not qualify for driving

If ... Else If ... Statement

The if ... else if ... statement is a more complex version of if else statement that maps it out for multiple conditions.

Syntax

If (expression 1)

{Statements that should be executed if the expression 1 is true }

Else if (expression 2)

{Statements to be executed when expression 2 is true}

Else if (expression 3)

{Statements to be executed when expression 3 is true }

Else if (expression 4)

{Statements to be executed when expression 4 is true }

Else

{Statements to be executed when all expressions are false}

This concept is similar to the if and if else statements. Whenever the if statements are true the statement below it is executed and when everything is false it executes the last statement under else.

EXAMPLE PROGRAM:

<html>

<body>

<script type="text/javascript">

<!--

var book = "math";

```

if( book == "history" ){
    document.write("<b>History Book</b>");
}

else if( book == "math" ){
    document.write("<b>Math Book</b>");
}

else if( book == "economics" ){
    document.write("<b>Economics Book</b>");
}

else{
    document.write("<b>Unknown Book</b>");
}
//-->
</script>

```

<p>Set the variable to different value and then try...</p>

</body>

<html>

OUTPUT

Math Book

Switch Case

Switch case is a simpler method of executing the multiple if.. else if... statements that was mentioned above. It is a method to perform multi way branch. The one downside to this is that the conditions are all dependent on the value of a single variable.

From JavaScript 1.2 edition, the switch statement were used in case of nested if statements. This is more efficient and handles it better than the if conditions.

Syntax

Switch (expression)

{

Case condition 1 : Statement

break;

Case condition 2 : Statement

break;

Case condition 3 : Statement

break;

Case condition 4 : Statement

break;

Case condition 5 : Statement

break;

Case condition 6 : Statement

break;

Case condition 7 : Statement

break;

default : statement

break

}

The break statement is often used to help the program to exit the switch case statements and proceed with the main program once the execution is done.

EXAMPLE PROGRAM

<html>

<body>

<script type="text/javascript">

<!--

var grade='A';

*document.write("Entering switch block
");*

switch (grade)

{

*case 'A': document.write("Good job
");*

break;

*case 'B': document.write("Pretty good
");*

break;

*case 'C': document.write("Passed
");*

break;

```

    case 'D': document.write("Not so good<br />");
    break;

    case 'F': document.write("Failed<br />");
    break;

    default: document.write("Unknown grade<br />")
}
document.write("Exiting switch block");
//-->
</script>

</body>
</html>

```

OUTPUT

```

Entering switch block
Good job
Exiting switch block

```

Loops

Like it was mentioned before there will be times when you will have to write the same statements over and over again to execute certain operations. In order to reduce the space and decrease the complexity of program, a concept known as the loops are introduced. JavaScript uses all the necessary loops required to reduce the number of line taken to write a program.

The While Loop

The while loop is the most basic loop in JavaScript that is also easy to execute. The purpose of a while loop is that, you can repeat a set of statements or blocks till the time it is required. This repetition will continue only if the while statement is true. The loop will cease as soon as it reaches false condition.

Syntax

The syntax of while loop is –

While (expression)

```

{
Statements that need to be executed until the expression remains true.
}

```

EXAMPLE PROGRAM:

```

<html>
  <body>

```

```

<script type="text/javascript">
  <!--
    var count = 0;
    document.write("Starting Loop ");

    while (count < 10){
      document.write("Current Count : " + count + "<br />");
      count++;
    }

    document.write("Loop stopped!");
  //-->
</script>

```

```

</body>

```

```

</html>

```

OUTPUT:

Starting Loop Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Current Count : 9

Loop stopped!

The Do ... While Loop

It is a more advanced version of while loop. The do while loop condition will execute once before the condition is rechecked and iterate. The loop will continue to execute if the while condition is true and will stop once it reaches false condition.

Syntax

The syntax of do.. while is

Do

{statements that need to be executed until the while condition remains true

} while (expression);

NOTE: Do not forget to place the semicolon after the while condition or else the loop will not execute properly.

EXAMPLE PROGRAM

```

<html>

```

```

<body>
  <script type = "text/javascript">
<!--
  Var count = 0;

  Document.write (" starting Loop" + "<br />");
  Do{

  Document.write ("Current Count : " + count + "<br/>");
  Count ++;
}
  While (count < 5);
  Document.write(" Loop stopped!");
  //-->
</script>

```

OUTPUT

```

Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop Stopped!

```

For Loop

The most compact form of looping is the For loop and it includes the 3 important parts which are listed below:

- Loop Initialization is where we initialize our counter to a starting value and the initialization statement in the loop begins.
- Test statement is similar to the if condition statement. This is the condition that determines when the loop should terminate. This condition is checked after each and every iteration.
- The iteration statement is usually a counter. This keeps the number of repetitions in check.

You have to combine all of these three parts in a statement which is separated using semi colons.

Syntax

The syntax for the for loop is described below –

```
For ( initialization ; test condition; iteration statement )
```

```
{ statements that need to be executed whenever the test condition is true }
```

EXAMPLE PROGRAM:

```
<html>
```

```
<body>
```

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

```
<!--
```

```
var count;
```

```
document.write("Starting Loop" + "<br />");
```

```
for(count = 0; count < 10; count++){
```

```
document.write("Current Count : " + count );
```

```
document.write("<br />");
```

```
}
```

```
document.write("Loop stopped!");
```

```
//-->
```

```
</script>
```

```
</body>
```

```
</html>
```

OUTPUT

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Current Count : 9

Loop stopped !

Conclusion

Thank you once again for choosing this book to learn JavaScript. JavaScript is a very important programming language that is essential in the structuring of a website or a web page. This language has gone through many revisions and the latest one also supports static and dynamic web browsing which is possible due to Perl programming language. One of the biggest merits of using JavaScript is that its programming structure is similar to all the other object oriented programming languages such as C language, C++ and Java. Though there are a lot more programming languages that are popping up in every corner, JavaScript acts as a keystone in web development without which the internet would be an extremely disorganized place.

I hope this book gives you an insight to the introduction and history of JavaScript that is helpful in learning about its evolution and where it is today. This book only covers basic programming where you will be familiar with the basic terminology and its usage. Learning these concepts will not only be helpful in learning JavaScript, but also in other programming languages since these are the building blocks of any structured programming.

Hope you have understood the basic concepts that deals with JavaScript and the specifications that is unique to it. You now have a basic understanding of the variables, keywords, operators and loops that merely just scratches the surface of what you can actually accomplish with JavaScript. There are more advanced concepts where you can create user defined data types known as objects or classes, then there are functions that can be conjured as and when you require it instead of repeating the same instructions over and over again and so many more real time applications such as cookies, events, dialog boxes and page printing that is possible with JavaScript.

Now that you have some knowledge on what to program and where to program it, I encourage you to try it by yourself in any of the methods that I have mentioned above. Keep trying and keep learning since this could help you better your career in ways that you didn't even think was possible!

Happy programming!

Thank you for Reading! I Need Your Help...

Dear Reader,

I Hope you Enjoyed “**JavaScript: QuickStart Guide - Effective JavaScript Programming**

”. I have to tell you, as an Author, I love feedback! I am always seeking ways to improve my current books and make the next ones better. It's readers like you who have the biggest impact on a book's success and development! So, tell me what you liked, what you loved, and even what you hated. I would love to hear from you, and I would like to ask you a favor, if you are so inclined, would you please share a minute to review my book. Loved it, Hated it - I'd just enjoy your feedback. As you May have gleaned from my books, reviews can be tough to come by these days and You the reader have the power make or break the success of a book. If you'd be so kind to [CLICK HERE](#) to review the book, I would greatly appreciate it! Thank you so much again for reading “**JavaScript: QuickStart Guide - Effective JavaScript Programming**” and for

spending time with me! I will see you in the next one!

Check Out More From The Publisher...

HTML: QuickStart Guide Creating an Effective Website

by William Fischer

<http://www.amazon.com/HTML-QuickStart-Effective-Wordpress-Javascript-ebook/dp/B01ACB7OMS>

SEO: Marketing Strategies to Dominate the First Page

by Grant Kennedy

<http://www.amazon.com/SEO-Marketing-Strategies-analytics-optimization-ebook/dp/B01ACB7LQM>

CSS :

Quick Start Guide - Effective Web Design

© Copyright 2016 - All rights reserved.

In no way is it legal to reproduce, duplicate, or transmit any part of this document in either electronic means or in printed format. Recording of this publication is strictly prohibited and any storage of this document is not allowed unless with written permission from the publisher. All rights reserved.

The information provided herein is stated to be truthful and consistent, in that any liability, in terms of inattention or otherwise, by any usage or abuse of any policies, processes, or directions contained within is the solitary and utter responsibility of the recipient reader. Under no circumstances will any legal responsibility or blame be held against the publisher for any reparation, damages, or monetary loss due to the information herein, either directly or indirectly.

Respective authors own all copyrights not held by the publisher.

Legal Notice:

This book is copyright protected. This is only for personal use. You cannot amend, distribute, sell, use, quote or paraphrase any part or the content within this book without the consent of the author or copyright owner. Legal action will be pursued if this is breached.

Disclaimer Notice:

Please note the information contained within this document is for educational and entertainment purposes only. Every attempt has been made to provide accurate, up to date and reliable complete information. No warranties of any kind are expressed or implied. Readers acknowledge that the author is not engaging in the rendering of legal, financial, medical or professional advice.

By reading this document, the reader agrees that under no circumstances are we responsible for any losses, direct or indirect, which are incurred as a result of the use of information contained within this document, including, but not limited to, —errors, omissions, or inaccuracies.

Contents

Introduction

Prerequisites

Chapter 1: Basic Tags in HTML

Heading Tags

Paragraph Tag

Line Break Tag

Centering Content

Horizontal Lines

Preserve Formatting

Nonbreaking Spaces

Chapter 2: HTML Elements

HTML Tag vs. Element

Nested Elements in HTML

Attributes

Internationalization Attributes

Chapter 3: Formatting in HTML

Bold Text

Italic Text

Underlined Text

Strike Text

Monospaced Font

Superscript Text

Subscript Text

Inserted Text

Deleted Text

Larger Text

Smaller Text

Grouping Content

Chapter 4: HTML Phrase Tags

Emphasized Text

Marked Text

Strong Text

Text Abbreviation

Acronym Element

Text Direction

Special Terms

Quoting Text

Short Quotations

Text Citations

Computer Code

Keyboard Text

Programming Variables

Program Output

Address Text

Chapter 5: Meta Tags

Specifying Keywords

Document Description

- Document Revision Date
- Document Refreshing
- Page Redirection
- Setting Cookies
- Setting Author Name
- Specify Character Set

Chapter 6: Comments

- Valid vs. Invalid Comments
- Multiline Comments
- Conditional Comments
- Using Comment Tag
- Commenting Script Code
- Commenting Style Sheets

Chapter 7: HTML Images

- Insert Image
- Set Image Location
- Set Image Width/Height
- Set Image Border
- Set Image Alignment

Chapter 8: HTML Lists

- HTML Unordered Lists
- HTML Ordered Lists

Chapter 9: HTML Colors

- HTML Color Coding Methods
- HTML Colors - Color Names
- HTML Colors - Hex Codes
- HTML Colors - RGB Values
- Browser Safe Colors

Conclusion

Introduction

Chapter 1: Introduction to JavaScript Programming

- History

Chapter 2: JavaScript - Specifications and Features

Chapter 3: JavaScript – Syntax

Chapter 4: Enabling and Disabling JavaScript in your Web Browser

Chapter 5: JavaScript Data Types

Chapter 6: Operators

Chapter 7: Case Statements and Loops

Conclusion

Introduction

Chapter 1: The Advantages of Using CSS

Chapter 2: CSS Modules

Chapter 3: Inclusion

Chapter 4: Measurement Units

Chapter 5: Colors

Chapter 6: Background

Chapter 7: Fonts

Chapter 8: Text

Chapter 9: Using Images

Chapter 10: Links

Chapter 11: Tables

Chapter 12: Borders

Chapter 13: Margins

Conclusion

Introduction

CSS, the shortened version of Cascading Style Sheets, is one of the simplest of all the design languages, the one you need to use for the most effective web design. CSS will handle how your webpages look and how they feel; you can set the text color, the font style, how much space there is between each paragraph, how the web page columns are laid out and sized, the colors and images needed for the background and all the other layout and design variations that you need to make your website work on all different screen sizes and devices.

CSS is incredibly easy to understand and learn and it gives you powerful control over how your HTML document are presented. It is commonly combines with two of the more popular markup languages, HTML or XHTML, to provide you with all the tools you need to produce a well-designed website.

For the purpose of this book, I am going to take you through all of the main aspects of web design that you can control CSS, together with sample code. I hope that, by the end of it, you have some idea of how easy CSS is to use and the amazing results you can produce.

Chapter 1: The Advantages of Using CSS

CSS has a lot of advantages over and above any other similar program, including:

Time-Savings - with CSS, you can write it once and then use the exact same sheet in a number of different HTML pages. Each HTML element can have its own separate style that can be applied to multiple web pages.

Faster Page Load Times – when you use CSS, there is no need to write the HTML tag attribute every single time. Instead, you can write just one rule for a tag and then apply it to every occurrence of the tag. Less code equals much faster loading times

Easy to Maintain - when you want to make a global change, all you need to do is change the style; each element relating to that style in all the web pages it was used in will be changed

Superior to HTML in Style – CSS provides access to a much bigger range of attributes than HTML does so your HTML page will look better than if you used HTML attributes.

Compatibility with Multiple Devices – Your style sheets allow for content to be optimized for more multiple types of device. Using the same HTML document, you can produce different website versions for printing, for PDAs or for mobile devices, for example.

Global Web Standards – More and more, CS I being recommended over HTML so now is a good time to start learning it and using it in all your HTML pages. That way, they will be compatible with the browsers of the future.

Offline Browsing CSS is able to store applications locally using offline cache. This mean that you will find it easier to view offline websites and, because of the cache, they will load faster and the website will perform better.

As you can see, CSS offers a ton of advantages over using HTML, not least the fact that it is now the recognized standard. With CSS becoming more and more poplar, now is the best time to start your learning process.

Without any further ado, let us delve into the CSS elements that you use for web design.

Before I go any further, I must state that some of the example code used in the book is attributed to www.tutorialspoint.com.

Chapter 2: CSS Modules

CSS modules take on the existing CSS specifications along with a number of extension features, including:

- Selectors
- Box Model
- Image Values
- Backgrounds and borders
- Replaced content
- Text effect
- D and 3D transformations
- Animations
- Multiple column layout
- Use Interface

CSS is made up of a number of style rules that are interpreted by the browser before being applied to the right element of the document. Style rules are split into three:

Selector - this is the HTML tag that the style is applied to. Examples of these tags are, **<h1>** or **<table>**.

- Property – Properties are types of HTML ta attributes. In simple terms, all HTML attributes are changed to CSS properties, for example, color border, etc.
- Value – Each property is assigned a value or value. For example, the color property may have a value of red or perhaps #F2F2F2, etc.

The syntax for a CSS style rule is:

```
selector { property: value }
```

For example, you could define a border on a table as:

```
table{ border :1px solid #C00; }
```

In this example, the table is a selector and border is a property. It has been given a value of 1px solid #C00

Selectors can be defined in a number of ways so let's look at each different one:

Type Selectors

You saw an example of the Type selector above but let's have a look at another - a way of adding a color to all of the level 1 headings:

```
h1 {  
color: #36CFFF;  
}
```

Universal Selectors

Instead of selecting the elements that relate to a specific type, this selector just matches the names of the element types. For example:

```
{  
color: #000000;  
}
```

In this example, the content of all the elements has been turned to black

Descendant Selectors

Let's say you want to give a particular element a style but only when it is inside a specific element. In the following example, we are applying the style rule to the `` element but only when it is outside of the `` tag:

```
ul em {  
color: #000000;  
}
```

Class Selectors

Class selectors allow you to use the class attribute of the elements to define your style rules. All of the elements of that particular class are formatted as per the defined rule:

```
.black {  
color: #000000;  
}
```

In this example, the rule has turned the content of all the elements with the class attribute of black to the color black. You could go one step further if you like:

```
h1.black {  
color: #000000;  
}
```

Now you have specified that only the content inside the `<h1>` element with a class attribute of black will be turned to black.

You are not limited to just one class selector for an element though. Look at this example:

```
<p class="center bold">
```

The style of this paragraph is defined by the classes Bold and Center

```
</p>
```

ID Selectors

Style rules can be defined on the id attribute of an element and, when a rule is defined, it will format all elements with the specified id attribute:

```
#black {  
color: #000000;  
}
```

This rule will turn the content of all elements that has the id attribute to black. You can be a bit more specific if you like, for example:

```
h1#black {  
color: #000000;  
}
```

This rule turns the content for `<h1>` elements to black ONLY if the id attribute is set to black.

The real power of an id selector become known when they are used as foundations for the descendant selector, for example:

```
#black h2 {  
color: #000000;  
}
```

Here, you can see that all of the level 2 headings will be turned to black only when they lie inside a tag that has the id attribute set as black

Child Selectors

The child selector is similar to the descendant selector but with different functionality. Look at this example:

```
body > p {  
color: #000000;  
}
```

This will turn the paragraphs to black only if they are the direct child of the <body> element. Other paragraphs that are inside other elements, like <td> or <div> would not change.

Attribute Selectors

Styles can also be applied to HTML element that have specific attributes. Look at the example below; you can see that the style rule is going to match all the input elements that have a type attribute with a text value:

```
input [type = "text"]{  
color: #000000;  
}
```

The advantage to using this method is that <input type = "submit" /> is not affected and the color change only applied to certain text fields.

The following rules are for the attribute selector:

- p[lang] - this selects all of the paragraph elements that have a lang attribute
- p[lang]= "fr" – this selects all the paragraph elements with lang attributes that have an exact value of "fr"
- p[lang~="fr"] – this selects all the paragraph elements that have a lang attribute containing "fr"
- p[lang|="en"] – this selects all the paragraph elements with a lang attribute containing exact values of "en" or that start with "en-".

Multiple Style Rules

Multiple style rules can be defined for one single element and these should be defined to combine a number of properties and their corresponding value into one block, as in this example:

```
h1 {  
  
color: #36C;  
  
font-weight: normal;
```

```
letter-spacing: .4em;  
margin-bottom: 1em;  
text-transform: lowercase;  
}
```

In this example, you can see that the pairs of properties and their value are each separated by a semi-colon (;). They can stay in one single line or you can spread them over multiple lines but, to read them better, we have kept them on single lines.

Grouping Selectors

Styles can be applied to multiple selectors. You just have to remember to separate each with a comma, as shown in this example:

```
h1, h2, h3 {  
    color: #36C;  
    font-weight: normal;  
    letter-spacing: .4em;  
    margin-bottom: 1em;  
    text-transform: lowercase;  
}
```

This style rule will apply to all of the h1, h2 and h3 elements as well. The order in the list is not relevant because all of the elements in the selector are going to have the declarations applied to them. As you can see from the example below, various class selectors can be combined:

```
#content, #footer, #supplement {  
    position: absolute;  
    left: 510px;  
    width: 200px;  
}
```


Chapter 3: Inclusion

You have four ways of associating styles with HTML documents, the most common of which is the inline CSS and the External CSS methods.

Embedded CSS

CSS rules can be put into an HTML document with the use of the <style> element. The tag goes inside the <head>...</head> Tag and the rules that are defined with this syntax are applicable to all of the available elements. The following example shows an embedded CSS style element:

```
<!DOCTYPE html>
<html>
  <head>

    <style type = "text/css" media = "all">
      body {
        background-color: linen;
      }
      h1 {
        color: maroon;
        margin-left: 40px;
      }
    </style>

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

Attributes

The attributes commonly associated with the <style> elements are shown in the following table:

Attribute	Value	Description
type	Text/css	This attribute specifies the language for the style sheet as an MIME content type and is a required attribute
media	screen tty	This specifies which device the document is going to be displayed on. The default value is all and this is

tv	classed as an optional attribute
projection	
handheld	
print	
braille	
aural	
all	

Inline CSS

You can also use the style attributes from any HTML elements to define the rules. These specific rules will only be applied that the particular element. The generic syntax is:

```
<element style = "...style rules....">
```

Attributes

The value of a style attribute is a mixture of different style declarations, each separated by semi colons. The following example is based on the generic syntax:

```
<html>
  <head>
  </head>
  <body>
    <h1 style = "color:#36C;" > This is inline CSS </h1>
  </body>
</html>
```

External CSS - The <link> Element

You can use the <link> element as a way of including an external stylesheet in the HTML document. External stylesheet are separate text files that have a .css extension. You will define all of your style rules in this file and this can then be added to any HTML document by using the <link element. The generic syntax is:

```
<head>
  <link type = "text/css" href = "... " media = "... " />
</head>
```

Attributes

The attributes that are associated with the <style> elements are as follows:

Attribute	Value	Description
type	text/css	This specifies the language for the style sheet as a MIME content type and is a required attribute
href	URL	This specifies that the style sheet has style rules and is also a required attribute
media	Screen	Tis specifies which devices the document can

tty	be displayed on and is an optional attribute.
tv	The default value is All
projection	
handheld	
print	
braille	
aural	
all	

As an example, consider a style sheet called mystyle.css with these rules:

```
h1, h2, h3 {
  color: #36C;
  font-weight: normal;
  letter-spacing: .4em;
  margin-bottom: 1em;
  text-transform: lowercase;
}
```

Now, using the following code, you can add this file to any HTML document:

```
<head>
  <link type = "text/css" href = "mystyle.css" media = " all" />
</head>
```

Imported CSS - @import Rule

The @import rule is used to import external stylesheets in a similar way to the <link> element. The generic syntax is:

```
<head>
  <@import "URL";
</head>
```

In this syntax, URL is used as the URL of the specific stylesheet that contains the style rules. Another way of writing it would be:

```
<head>
  <@import url("URL");
</head>
```

The following example shows you how to import the style sheet to the HTML document:

```
<head>
  @import "mystyle.css";
</head>
```

Overriding CSS Rules

So far, we have looked at four different ways to include a style sheet in an HTML document. Now we look at how to override any of the style sheet rules:

- An inline style sheet will take the top priority, including over any rule that is defined in an external sheet or defined in `<style> ... </style>` tags
- Any `<style> ... </style>` tag will override an external sheet file
- External style sheet rules are the lowest priority and will only be applied if the inline style or `<style> ... </style>` tags do not apply

How to Handle an Old Browser

There are quite a few of the older browsers that don't provide support for CSS so, when you are writing your Embedded CSS in HTML documents, please take care. The following example show how to use comment tags as a way of hiding CSS from an older browser:

```
<style type="text/css">
```

```
<!--
```

```
body, td {
```

```
color: blue;
```

```
}
```

```
-->
```

```
</style>
```

CSS Comments

Sometimes you will need to add extra comments in to your style sheet blocks but it is very easy to comment on any part of a style sheet. All you do is put your comments within `/*.... this is a comment in a style sheet.... */`.

You can also use `/*.... */` to comment on multi-line blocks in the same way that you would in C or C++ languages.

For example:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
p {
```

```
color: red;
```

```
/* a single-line comment */
```

```
text-align: center;
```

```
}
```

```
/* a multi-line comment */
```

```
</style>
```

```
</head>
```

```
<body>
```

```
  <p>Hello World!</p>
```

```
</body>
```

```
</html>
```

This will result in “Hello, World!” being displayed on your screen.

Chapter 4: Measurement Units

CSS supports several different measurements including relative measures, like percentages, and absolute unit, like centimeters, inches, points, etc. These values are required when you specify measurements in the Style rules, for example, e.g. border = "1px solid red".

The following table lists the CSS measurement units together with examples:

Unit	Description	Example
%	This defines the measurement as a percentage relative to other values. A typical one would be an enclosing element.	p {font-size: 16pt; line-height: 125%;}
cm	This defines measurements in centimeters	div {margin-bottom: 2cm;}
em	This is a relative measurement for font height in em spaces. An em unit is the equivalent of the size of any given font and, because of this, if you were to assign 12 pt to a font, each of the em units would be 12 pt. This means that 2em would equal 24 pt.	p {letter-spacing: 7em;}
ex	This defines the measurement that is relative to the x-height of the font. We determine the x-height by the height of the lowercase x in the specified font	p {font-size: 24pt; line-height: 3ex;}
in	This defines the measurement in inches	p {word-spacing: .15in;}
mm	This defines the measurement in millimeters	p {word-spacing: 15mm;}
pc	This defines the measurement in picas, which is equivalent to 12 points. This means that there are 6 picas to an inch.	p {font-size: 20pc;}
pt	This define the measurement in points, which is defined as 1/72 of an inch	body {font-size: 18pt;}
px	Defines the measurement in screen pixels.	p {padding: 25px;}
vh	This is 1% of the viewport height.	h2 {font-size: 3.0vh; }

vw This is 1% of the viewport width

```
h1 {  
font-size:  
5.9vw; }
```

vmin This is 1vw or 1vh, whichever is the smallest

```
p { font-size:  
2vmin; }
```


Chapter 5: Colors

In CSS we use values to specify colors. Usually these are used when we set the color for the text of an element, which is the foreground, or the background of the element. We can also use them to change the color of decorative effects, such as a border.

The color values can be specified in several different formats, as displayed in the following table:










Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%);}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

Let's look at these formats in more detail:

Hex Codes

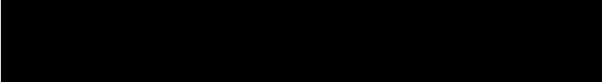







Hexadecimals are 6-digit color representations. The first digit – RR – represent red values, GG represents green values and BB represents the blue values.

The values can be taken from graphics software, such as JASC Paintshop Pro, Adobe Photoshop, Advanced Paintbrush, etc. Each of the hexadecimal codes are preceded by a # or a £ sign and the following are examples of hexadecimal notations:

Color	Color HEX
	#000000
	#FF0000
	#00FF00
	#0000FF
	#FFFF00
	#00FFFF
	#FF00FF
	#C0C0C0
	#FFFFFF

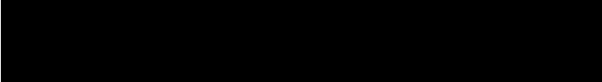








Short Hex Codes

This is just a shorter version of the hex code, and each digit is replicated so that you get the equivalent 6-digit value. These values can also be taken from graphics software and are also preceded by the # or £ sign:

Color	Color HEX
	#000
	#F00
	#0F0
	#0FF
	#FF0
	#0FF
	#F0F
	#FFF

RGB Values

We specify this value with the `rgb()` property, which takes three values – red, green and blue. Values are percentages or integers between 0 and 255. Be aware that not all browsers offer support for `rgb()` so try not to use it.

Color	Color RGB
	<code>rgb(0,0,0)</code>
	<code>rgb(255,0,0)</code>
	<code>rgb(0,255,0)</code>
	<code>rgb(0,0,255)</code>
	<code>rgb(255,255,0)</code>
	<code>rgb(0,255,255)</code>
	<code>rgb(255,0,255)</code>
	<code>rgb(192,192,192)</code>
	<code>rgb(255,255,255)</code>

Building the Color Codes

If you use the CSS Color Code Builder, you can build, quite literally, millions of different color code but you will need a browser that is Java-enabled to do this.

Browser-Safe Colors

The following tables show you the 216 colors that are considered to be safe and are computer-independent. They vary from hexa code 000000 through to FFFFFFFF and are safe because all computers can display the colors in the right way on a 25 color palette:

	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF

CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

Chapter 6: Background

In this chapter, we are going to look at setting backgrounds for different HTML elements. The following properties can be set:

- background-color - to set the background color of the element
- background-image - to set the background image of the element
- background-repeat – to control background image repetition
- background-position – to control where a background image is positioned
- background-attachment – to control how a background image scrolls
- background – used as a shorthand method of specifying various background properties

Setting the Background Color

The following example shows you how to set background color on an element:

```
<html>
<head>
<body>
  <p style = "background-color:yellow;">
    This text has a yellow background color.</p>
</body>
</head>
```

```
<html>
```

You will see the following on your screen:

The text has a yellow background color.

How to Set the Background Image

The following example shows how to call a local stored image to set the background:

```
<html>
<head>
  <style>
    body {
      background-image: url("/css/images/css.jpg");
      background-color: #cccccc;
    }
  </style>
<body>
  <h1>Hello World!</h1>
</body>
</head>
```

<html>

You will now see "Hello, World!" appear on the specified background

How to Repeat the Background Image:

The following is an example of how to repeat a small background image. The no-repeat value can be used for the background-repeat property if the image is NOT to be repeated and is only to be displayed once. By default, the value of background-repeat is set to repeat:

<html>

<head>

<style>

body {

background-image: url("/css/images/css.jpg");

background-repeat: repeat;

}

</style>

</head>

<body>

<p>Tutorials point</p>

</body>

</html>

The background that we used in the last example will now be shown repeated. In the following example, we can see how to repeat an image vertically:

<html>

<head>

<style>

body {

background-image: url("/css/images/css.jpg");

background-repeat: repeat-y;

}

</style>

</head>

<body>

<p>Tutorials point</>

</body>

</html>

The chosen background image will be repeated vertically.

In the next example, we look at horizontal repetition:

```
<html>
  <head>
    <style>
      body {
        background-image: url("/css/images/css.jpg");
        background-repeat: repeat-x;
      }
    </style>
  </head>
  <body>
    <p>Tutorials point</p>
  </body>
</html>
```

The chosen background image will be repeated horizontally

How to Set the Position of the Background Image

This example shows you how to set the image in a position that is 100 pixels in from the left hand side:

```
<html>
  <head>
    <style>
      body {
        background-image: url("/css/images/css.jpg");
        background-position:100px;
      }
    </style>
  </head>
  <body>
    <p>Tutorials point</p>
  </body>
</html>
```

The background image will be placed in the specified position

Now we look at how to set the image 100 pixels in from the left and 200 pixels from the top:

```
<html>
```

```
<head>
  <style>
    body {
      background-image: url("/css/images/css.jpg");
      background-position:100px 200px;
    }
  </style>
</head>
<body>
  <p>Tutorials point</p>
</body>
</html>
```

The background image will be placed in the specified position

How to Set the Background Attachment

This determines if an image is to be fixed or if it should scroll when the page is scrolled. First, we look at a fixed image:

```
<!DOCTYPE html>
<html>
  <head>

    <style>
      body {
        background-image: url('/css/images/css.jpg');
        background-repeat: no-repeat;
        background-attachment: fixed;
      }
    </style>

  </head>
  <body>

    <p>The background-image is fixed. Try scrolling down the page.</p>
    <p>The background-image is fixed. Try scrolling down the page.</p>
    <p>The background-image is fixed. Try scrolling down the page.</p>
```

```
<p>The background-image is fixed. Try scrolling down the page.</p>
<p>The background-image is fixed. Try scrolling down the page.</p>
<p>The background-image is fixed. Try scrolling down the page.</p>
<p>The background-image is fixed. Try scrolling down the page.</p>
<p>The background-image is fixed. Try scrolling down the page.</p>
<p>The background-image is fixed. Try scrolling down the page.</p>
```

```
</body>
```

```
</html>
```

You will see this line repeated however many times it is in the code:

The background-image is fixed. Try scrolling down the page.

Followed by this:

If you do not see any scrollbars, try to resize the browser window.

Next, we look at setting a scrolling background:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
  body {
```

```
    background-image: url('/css/images/css.jpg');
```

```
    background-repeat: no-repeat;
```

```
    background-attachment: fixed;
```

```
    background-attachment:scroll;
```

```
  }.
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<p>The background-image is fixed. Try scrolling down the page.</p>
```

```
<p>The background-image is fixed. Try scrolling down the page.</p>
```

```
<p>The background-image is fixed. Try scrolling down the page.</p>
```


`<p>The background-image is fixed. Try scrolling down the page.</p>`

`<p>The background-image is fixed. Try scrolling down the page.</p>`

`<p>The background-image is fixed. Try scrolling down the page.</p>`

`<p>The background-image is fixed. Try scrolling down the page.</p>`

`<p>The background-image is fixed. Try scrolling down the page.</p>`

`<p>The background-image is fixed. Try scrolling down the page.</p>`

`</body>`

`</html>`

Again, you will see this message repeated on a scrolling background

The background-image is fixed. Try scrolling down the page.

Followed by this:

If you do not see any scrollbars, try to resize the browser window.

How to use the Shorthand Property

The final example shows how to set all of the background properties at the same time by using the shorthand background property:

`<p style="background:url(/images/pattern1.gif) repeat fixed;">`

This paragraph has a fixed background image that will be repeated.

`</p>`

Chapter 7: Fonts

Now we are going to look at how to set up the fonts. The following are the font properties that you can set for an element:

- font-family - is used to change the font face
- font-style - is used to turn a font oblique or italic
- font-weight – is used to increase or decrease the boldness or lightness of a font
- font-variant – used to give the font a small caps effect
- font-size – is used to make a font bigger or smaller
- font – shorthand for specifying various font properties

How to Set the Font Family

The following example shows how to set the font family of a specific element. The value can be any font family name:

```
<html>
  <head>
</head>
  <body>
    <p style="font-family:georgia,garamond,serif;">
      This text will be rendered in georgia, garamond, or the default serif font
      depending on which ones are in your system.
    </p>
  </body>
</html>
```

It will produce the following result –

This text will be rendered in georgia, garamond , or the default serif font, depending on which ones are in your system

How to Set the Font Style

The following example shows how to set the style of an element. Values can be italic, normal, oblique, etc.

```
<html>
  <head>
</head>
  <body>
    <p style="font-style:italic;">
      This text is rendered in italic style
    </p>
  </body>
```

```
</html>
```

You will get the following result:

This text is rendered in italic style

How to Set the Font Variant

This example shows how to set the font variant with possible values of small-caps, normal, etc.

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style="font-variant:small-caps;">
```

```
The text is going to be rendered as small caps
```

```
</p>
```

```
</body>
```

```
</html>
```

You will get the following result:

THE TEXT IS GOING TO BE RENDERED AS SMALL CAPS

How to Set the Font Weight

The next example shows how we set the font weight of an element. This property will specify the bold level of the font and possible values include – normal, lighter, bold, bolder, 100, 200, 300, 400, 500, 600, 700, 800, 900, etc.

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style="font-weight:bold;">The font is bold.</p>
```

```
<p style="font-weight:bolder;">The font is bolder.</p>
```

```
<p style="font-weight:500;">The font is 500 weight.</p>
```

```
</body>
```

```
</html>
```

You will get the following result:

The font is bold.

The font is bolder.

The font is 500 weight.

How to Set the Font Size

This example shows how to set up the font size of the element. This is used as a way of controlling the font size and has possible values of – small, x-small, xx-small, xx-large, x-large, large, smaller, larger, %, or size in pixels:

```
<html>
  <head>
</head>
  <body>
    <p style="font-size:20px;">The font size is 20 pixels</p>
    <p style="font-size:small;">The font size is small</p>
    <p style="font-size:large;">The font size is large</p>
  </body>
</html>
```

You will get the following result:

The font size is 20 pixels

The font size is small

The font size is large

How to Set Font Size Adjust

This example will demonstrate how we set the font size adjust and lets you change the x-height to make the font more readable. The possible values can be any number:

```
<html>
  <head>
</head>
  <body>
    <p style="font-size-adjust:0.61;">
      The text uses a font-size-adjust value.
    </p>
  </body>
</html>
```

You will get the following result -

THE TEXT USES A FONT-SIZE-ADJUST VALUE.

How to Set the Font Stretch

This next example shows you how to set up the font stretch and it relies on your computer to have either an expanded or a condensed version of the particular font. The values could possibly be – wider, normal, narrower, extra-condensed, ultra-condensed, condensed, semi-condensed, expanded, semi-expanded,

extra-expanded, ultra-expanded:

```
<html>
  <head>
  </head>
  <body>
    <p style="font-stretch:ultra-expanded;">
      If this doesn't look like it works, most likely your computer does not have a condensed or
expanded version of the font
    </p>
  </body>
</html>
```

It will produce the following result –

If this doesn't look like it works, most likely your computer does not have a condensed or expanded version of the font

How to Set the Shorthand Property

The final example shows you how to use the font property to set all of the font properties at the same time:

```
<html>
  <head>
  </head>
  <body>
    <p style="font:italic small-caps bold 15px georgia;">
      This will apply all of the properties on the text at the same time.
    </p>
  </body>
</html>
```

It will produce the following result –

THIS WILL APPLY ALL OF THE PROPERTIES ON THE TEXT AT THE SAME TIME.

Chapter 8: Text

This chapter is going to take a look at how we can manipulate text with the use of some CSS properties. The following text properties can be used on an element:

- color – used to set what color the text is
- direction – used to set the direction of the text
- letter-spacing – used to add in or remove spaces between letter in a word
- text-indent – used to indent the first line of a paragraph of text
- text-align – used to align the text in an entire document
- text-decoration – used to add decoration to text, such as underline, overline, and ~~strikethrough~~
- text-transform – used to either capitalize the text or convert between upper and lower case letters
- white-space – used to control text flow and formatting
- text-shadow – used to set a shadow around the text

How to Set the Text Color

This example shows how we set the color of the text, using values of any color name in any format that is valid

```
<html>
  <head>
</head>
  <body>
    <p style="color:red;">
      This text is shown in red.
    </p>
  </body>
</html>
```

You will see the following result -

This text is shown in red.

How to Set the Direction of the Text

The next example shows how we set the text direction, using values of ltr or rtl

```
<html>
  <head>
</head>
  <body>
    <p style="direction:rtl;">
      This text is going to be rendered from right to left
    </p>
  </body>
```

</html>

You will see this result -

This text is going to be rendered from right to left

How to Set the Spaces Between Characters

This example shows how we set up the spaces between the characters, using values of normal or number specifying space

<html>

<head>

</head>

<body>

<p style="letter-spacing:5px;">

This text will have spaces between letters.

</p>

</body>

</html>

You will see this result -

T h i s t e x t w i l l h a v e s p a c e s b e t w e e n l e t t e r s .

How to Set the Spaces Between Words

The next example shows how we set space in between words, using values of normal or number specifying space

<html>

<head>

</head>

<body>

<p style="word-spacing:5px;">

This text has spaces between words.

</p>

</body>

</html>

You will see the following result -

This text has spaces between words.

How to Set the Text Indent

The next example shows how we indent the first line of a new paragraph using values of number specifying indent space or %

<html>

```
<head>
</head>
<body>
  <p style="text-indent:1cm;">
    The text has the first line indented by 1cm and it will remain at
    this position. This is done using the CSS text-indent property.
  </p>
</body>
</html>
```

You will see this result -

The text has the first line indented by 1cm and it will remain at this position. This is done using the CSS text-indent property

How to Set up the Text Alignment

This example shows how we align text, using values of justify, left, right and center

```
<html>
  <head>
  </head>
  <body>
    <p style="text-align:right;">
      The text will be right aligned.
    </p>
    <p style="text-align:center;">
      The text will be center aligned.
    </p>
    <p style="text-align:left;">
      The text will be left aligned.
    </p>
  </body>
</html>
```

You will see the following result -

The text will be right aligned.

The text will be center aligned.

The text will be left aligned.

How to Decorate Text

Now we look at an example showing how we decorate text, using values of none, overline, underline, blink and strikethrough

```
<html>
  <head>
</head>
  <body>
    <p style="text-decoration:underline;">
      This text is underlined
    </p>

    <p style="text-decoration:line-through;">
      This text is struck through.
    </p>

    <p style="text-decoration:overline;">
      This text has an over line.
    </p>

    <p style="text-decoration:blink;">
      This text has a blinking effect
    </p>
  </body>
</html>
```

You will see the following result -

This text is underlined

~~This text is struck through.~~

This text has an over line.

This text has a blinking effect

How to Set Text Cases

This example shows how we set up text cases using values of non, uppercase, lowercase and capitalize

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style="text-transform:capitalize;">
```

This text will be capitalized

```
</p>
```

```
<p style="text-transform:uppercase;">
```

This text will be in uppercase

```
</p>
```

```
<p style="text-transform:lowercase;">
```

This text will be in lowercase

```
</p>
```

```
</body>
```

```
</html>
```

You will see the following result -

This Text Will Be Capitalized

THIS TEXT WILL BE IN UPPERCASE

this text will be in lowercase

How to Set Whitespace between Text

This example shows how to handle whitespace in an element, using values of nowrap, normal and pre

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style="white-space:pre;">
```

This text will have a line break and the white-space pre setting will tell the browser to honor it the same as the HTML pre tag.</p>

```
</body>
```

```
</html>
```

You will see the following result -

This text will have a line break and the white-space pre setting will tell the browser to honor it the same as

the HTML pre tag.</p>.

How to Set a Text Shadow

This final example shows how to set a shadow around a piece of text. Be aware that this may not have the support of every browser

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style="text-shadow:4px 4px 8px blue;">
```

Provided your browser will support CSS text-shadow property, the text will have a blue shadow

```
</p>
```

```
</body>
```

```
</html>
```

You will see the following result -

Provided your browser will support CSS text-shadow property, the text will have a blue shadow

Chapter 9: Using Images

Images are an important part of websites and, although you shouldn't use too many, those you do use should be of good quality and in the right place. CSS helps to control the display of these images by allowing you to set the following properties:

- Border – this is used to set the width of the border on the image
- Height – this is used to set the height of the image
- Width – this is used to set the width of the image
- -most-opacity - this used to determine the opacity of the image

Image Border

This property is used to set the width of the border on the image and can use values of length or %. For example:

```
<html>
  <head>
</head>
  <body>
    
    <br />
    
  </body>
</html>
```

The result will be a logo without a border and then a logo with a dotted red border

Image Height

This property is used to set how high an image is and can use values of length or %. When you use %, the value is applied to the box which the image is in:

```
<html>
  <head>
</head>
  <body>
    
    <br />
    
  </body>
</html>
```

The result will be two logos, exactly the same, but one taller than the other

Image Width

This is used to set the width of an image using value of % or length. As with height, using percentage applies

to the box the image is in

```
<html>
  <head>
</head>
  <body>
    
    <br />
    
  </body>
</html>
```

The result will be two identical logos but of different sizes

-Moz-Opacity

The final property is used to set up the opacity of the image and is used with Mozilla Firefox. For Internet Explorer, we use **filter:alpha (opacity:x)**, to come up with transparent images.

In Mozilla Firefox, in the property (**-moz-opacity:x**), x may hold a value of anywhere between 0.0 and 1.0. The lower the value, the more transparent the image. By the same token, in Internet Explorer (**filter:alpha(opacity=x)**), the value of x can be anywhere between 0 and 100. Again, the lower the value, the more transparent the image is.

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

The result will be a logo with a border of solid red.

Chapter 10: Links

In this chapter, we take a look at how to use CSS to set the properties of a hyper link. The following properties can be set:

- **:link** is used to signify unvisited hyperlinks.
- **:visited** is used to signify visited hyperlinks.
- **:hover** is used to signify elements that have the mouse pointer hovering over the top.
- **:active** is used to signify elements that are currently being clicked.

Normally, these properties are stored in the header of an HTML document. Remember the following rules for effective use:

- a:hover comes after a:link and a:visited in CSS definition
- a:active comes after a:hover, as shown in the next example:

```
<style type="text/css">
  a:link {color: #000000}
  a:visited {color: #006600}
  a:hover {color: #FFCC00}
  a:active {color: #FF00CC}
</style>
```

Next we will look at how these properties are used for different effects on hyperlinks.

Set the Color

This example shows how we set the color of a link using values that are of any color name in any format that is valid:

```
<html>
  <head>
    <style type="text/css">
      a:link {color:#000000}
    </style>
  </head>
  <body>
    <a href="">Link</a>
  </body>
</html>
```

The result will be a black link. Please note that none of these links are clickable; they are just for demonstration purposes.

[Link](#)

The Color of Visited Links

This example shows how to set the color of a visited link, again with values of any color name in any format that is valid:

```
<html>
  <head>
    <style type="text/css">
      a:visited {color: #006600}
    </style>
  </head>
  <body>
    <a href=""> link</a>
  </body>
</html>
```

The result will be the following link, black to start and green when clicked on.

[Link](#)

Color of Links with Mouse Hover

This example shows how to change the color of a link when we hover the mouse pointer over it:

```
<html>
  <head>
    <style type="text/css">
      a:hover {color: #FFCC00}
    </style>
  </head>
  <body>
    <a href="">Link</a>
  </body>
</html>
```

The result will be a black link that changes to yellow when the mouse is hovered over the top

[Link](#)

Color of Active Links

This example shows how we change the color of an active link:

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

```
<style type="text/css">
```

```
  a:active {color: #FF00CC}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
  <a href="">Link</a>
```

```
</body>
```

```
</html>
```

The result will be a black link that changes to pink when it is clicked on.

Link

Chapter 11: Tables

In this chapter, we are going to look at the properties we can set for HTML tables. The following properties can be set:

- **Border-collapse** – this specifies whether the cells of the table should maintain their own appearance or if the browser should control the borders that touch
- **Border-spacing** – this specifies the width of the space between table cells
- **Caption-side captions** – these are in the `<caption>` element and rendered, by default, above the table. They can be used to set the position of the table caption
- **Empty-cells** – this specifies whether there should be a border shown on empty cells
- **Table-layout** – this lets the browser speed up the table layout by using the first width property it finds for a column without having to load the table and then rendering it

Next we look at these properties with examples.

Border-Collapse

This property may have two values – collapse and separate. This example uses both:

```
<html>
```

```
<head>
```

```
<style type="text/css">
```

```
table.one {border-collapse:collapse;}
```

```
table.two {border-collapse:separate;}
```

```
td.a {
```

```
border-style:dotted;
```

```
border-width:3px;
```

```
border-color:#000000;
```

```
padding: 10px;
```

```
}
```

```
td.b {
```

```
border-style:solid;
```

```
border-width:3px;
```

```
border-color:#333333;
```

```
padding:10px;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<table class="one">
```

```
<caption>Collapse Border Example</caption>
```

```
<tr><td class="a1 Cell 1 Collapse Example"></td></tr>
```

```
<tr><td class="2"> Cell 2 Collapse Example</td></tr>
```

```
</table>
```

```
<br />
```

```
<table class="two">
```

```
<caption>Separate Border Example</caption>
```

```
<tr><td class="1"> Cell 1 Separate Example</td></tr>
```

```
<tr><td class="2"> Cell 2 Separate Example</td></tr>
```

```
</table>
```

```
</body>
```

```
</html>
```

This will give you these results;

Collapse Border:
Cell 1 Collapse Example
Cell 2 Collapse Example

Separate Border:
Cell 1 Separate Example
Cell 2 Separate Example

Border-Spacing

This property specifies how much distance there is between the border of two adjacent cells. It can take either of two values, each of which should be a length unit. One value applies to horizontal and vertical borders while using two values will refer to the vertical and the horizontal spacing.

```
<style type="text/css">
```

```
/* If you provide a value */
```

```
table.example {border-spacing:10px;}
/* This is how to provide two values */
table.example {border-spacing:10px; 15px;}
</style>
```

The next example modifies the first example:

```
<html>
<head>

<style type="text/css">
  table.one {
    border-collapse:separate;
    width:400px;
    border-spacing:10px;
  }
  table.two {
    border-collapse:separate;
    width:400px;
    border-spacing:10px 50px;
  }
</style>
```

```
</head>
```

```
<body>
```

```
<table class="one" border="1">
  <caption>Separate Border Example with border-spacing</caption>
  <tr><td> Cell A Collapse Example</td></tr>
  <tr><td> Cell B Collapse Example</td></tr>
</table>
<br />
```

```
<table class="two" border="1">
  <caption>Separate Border Example with border-spacing</caption>
  <tr><td> Cell A Separate Example</td></tr>
```

```
<tr><td> Cell B Separate Example</td></tr>
</table>
```

```
</body>
```

```
</html>
```

Caption-Side Property

This property allows you to specify the position of the <caption> element content in relation to the table. It can use these four properties – top, left, bottom, right. The following example uses all of the four values:

```
<html>
```

```
<head>
```

```
<style type="text/css">
```

```
caption.top {caption-side:top}
```

```
caption.bottom {caption-side:bottom}
```

```
caption.left {caption-side:left}
```

```
caption.right {caption-side:right}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<table style="width:400px; border:1px solid black;">
```

```
<caption class="top">
```

```
This caption will appear at the top
```

```
</caption>
```

```
<tr><td > Cell A</td></tr>
```

```
<tr><td > Cell B</td></tr>
```

```
</table>
```

```
<br />
```

```
<table style="width:400px; border:1px solid black;">
```

```
<caption class="bottom">
```

```
The caption appears at the bottom
```

```
</caption>
```

```
<tr><td > Cell A</td></tr>
<tr><td > Cell B</td></tr>
</table>
<br />
```

```
<table style="width:400px; border:1px solid black;">
  <caption class="left">
    The caption appears at the left
  </caption>
  <tr><td > Cell A</td></tr>
  <tr><td > Cell B</td></tr>
</table>
<br />
```

```
<table style="width:400px; border:1px solid black;">
  <caption class="right">
    The caption appears at the right
  </caption>
  <tr><td > Cell A</td></tr>
  <tr><td > Cell B</td></tr>
</table>
```

```
</body>
```

```
</html>
```

Empty-Cells

This property is used to indicate if an empty cell should have a border or not. It can have one of these three value – hide, show, inherit. This example shows the property being used to hide the borders of the empty cells in the <table> element:

```
<html>
```

```
<head>
```

```
<style type="text/css">
```

```
table.empty{
```

```
width:350px;
```

```
border-collapse:separate;
empty-cells:hide;
}
td.empty{
padding:5px;
border-style:solid;
border-width:1px;
border-color:#999999;
}
</style>
```

```
</head>
```

```
<body>
```

```
<table class="empty">
```

```
<tr>
```

```
<th></th>
```

```
<th>Title one</th>
```

```
<th>Title two</th>
```

```
</tr>
```

```
<tr>
```

```
<th>Row Title</th>
```

```
<td class="empty">value</td>
```

```
<td class="empty">value</td>
```

```
</tr>
```

```
<tr>
```

```
<th>Row Title</th>
```

```
<td class="empty">value</td>
```

```
<td class="empty"></td>
```

```
</tr>
```

```
</table>
```


</tr>

</table>

</body>

</html>

Chapter 12: Borders

Border properties allow you to say what the border of an element box should look like. The following are the three properties that you can control

- **Border-color** – allows you to specify the color of the border
- **Border-style** – allows you to specify the style from a choice of values, including a solid line, dashed line, double line, etc.
- **Border-width** – allows you to specify the width of the border

Border-Color

This property lets you change the color of a border surrounding an element. You can change each side of the border individually using these properties:

- **Border-bottom-color**
- **Border-top-color**
- **Border-left-color**
- **Border-right-color**

This example shows all four properties in use:

```
<html>
```

```
<head>
```

```
<style type="text/css">
```

```
  p.example1{
```

```
    border:1px solid;
```

```
    border-bottom-color:#009900; /* Green */
```

```
    border-top-color:#FF0000; /* Red */
```

```
    border-left-color:#330000; /* Black */
```

```
    border-right-color:#0000CC; /* Blue */
```

```
  }
```

```
  p.example2{
```

```
    border:1px solid;
```

```
    border-color:#009900; /* Green */
```

```
  }
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<p class="example1">
```

The example shows all borders in different colors.

```
</p>
```

```
<p class="example2">
```

The example shows all borders in green only.

```
</p>
```

```
</body>
```

```
</html>
```

The result will

This example shows all borders in different colors.

This example shows all borders in green only.

Border-style

This property let you select one of these styles:

- None – has no border at all
- Solid – gives the border a solid line
- Dotted – the border is made of dots
- Dashed – the border is made of short lines
- Double – this is two solid lines
- Groove – make the border look like it has been carved into the page
- Ridge – does the opposite of groove
- Inset – the box looks embedded in the page
- Outset – the box looks as though it is coming out of the page
- Hidden – exactly the same as for None but with the exception of a border-conflict resolution for table elements

As with color, you can change the style of each individual side of the border, using these properties:

- Border-bottom-style
- Border-top-style
- Border-left-style
- Border-right-style

This example uses all four of these styles:

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>.
```

```
<p style="border-width:4px; border-style:none;">
```

This is a border with a width of none.

</p>

<p style="border-width:4px; border-style:solid;">

This is a solid border.

</p>

<p style="border-width:4px; border-style:dashed;">

This is a dashed border.

</p>

<p style="border-width:4px; border-style:double;">

This is a double border.

</p>

<p style="border-width:4px; border-style:groove;">

This is a groove border.

</p>

<p style="border-width:4px; border-style:ridge">

This is a ridge border.

</p>

<p style="border-width:4px; border-style:inset;">

This is an inset border.

</p>

<p style="border-width:4px; border-style:outset;">

This is an outset border.

</p>

<p style="border-width:4px; border-style:hidden;">

This is a hidden border.

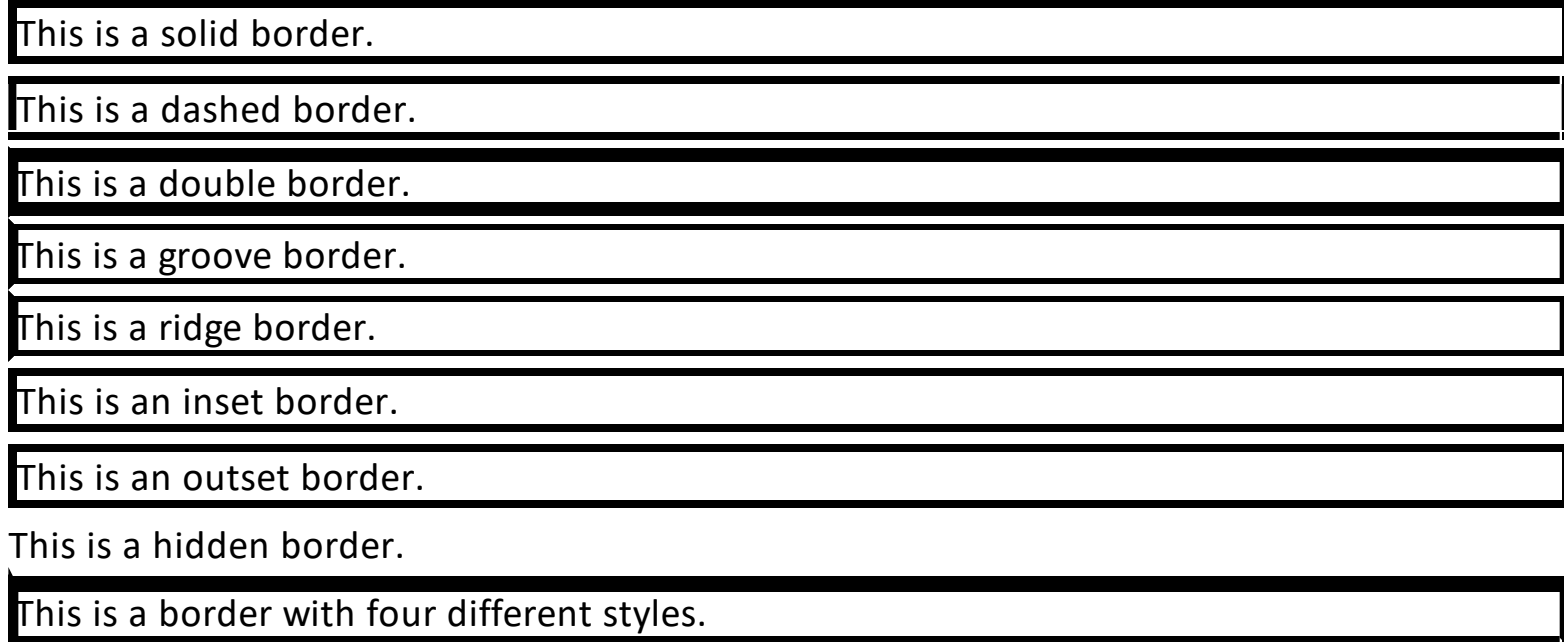
</p>

```
<p style="border-width:4px;border-top-style:solid;
border-bottom-style:dashed; border-left-style:groove; border-right-style:double;">
This is a border with four different styles.
</p>
</body>
```

```
</html>
```

The result will be -

This is a border with none width.



Border-Width

This allows you to set the width of the border, using a value of length in pt., px or cm or you can set it to thick, medium or thin. You can change each side of the border individually, using these properties:

- border-bottom-width
- border-top-width
- border-left-width
- border-right-width

This example use all of these properties

```
<html>
<head>
</head>
<body>
<p style="border-width:4px; border-style:solid;">
This is a solid border with a width of 4px.
</p>
```

```
<p style="border-width:4pt; border-style:solid;">
```

This is a solid border whose width is 4pt.

```
</p>
```

```
<p style="border-width:thin; border-style:solid;">
```

This is a solid border whose width is thin.

```
</p>
```

```
<p style="border-width:medium; border-style:solid;">
```

This is a solid border whose width is medium;

```
</p>
```

```
<p style="border-width:thick; border-style:solid;">
```

This is a solid border whose width is thick.

```
</p>
```

```
<p style="border-bottom-width:4px;border-top-width:10px;  
border-left-width: 2px;border-right-width:15px;border-style:solid;">
```

This is a border with four different widths.

```
</p>
```

```
</body>
```

```
</html>
```

The result will be:

This is a solid border whose width is 4px.

This is a solid border whose width is 4pt.

This is a solid border whose width is thin.

This is a solid border whose width is medium;

This is a solid border whose width is thick.

This is a border with four different widths.

Shorthand Border Properties

Using the shorthand border property, you can specify the style, color and width of the lines using just one property instead of three. The following example uses the border property to combine all three and you

will find that this is the most commonly used method of setting a border around an element:

```
<html>  
  <head>  
  </head>  
  <body>  
    <p style="border:4px solid red;">
```

Another example:

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


Chapter 13: Margins

The final chapter deals with the margin property, i.e. the space around the HTML element. Margin property values are not inherited by child elements and it is also possible that you can use negative value as a way of overlapping the content.

Remember – the vertical margins that are adjacent to one another (that is the top and bottom margins) are going to collapse into each other so that distance between each block is NOT the sum of the margins; instead it is the larger of the margins or the same as one margin if both are the same size.

The following properties can be used:

- Margin – specifies the shorthand for setting all the margin properties in one single declaration
- Margin-bottom – specifies the bottom margin
- Margin-top – specifies the top margin
- Margin-left – specifies the left margin
- Margin-right – specifies the right margin

Now we will look at how these properties are used.

Margin Property

This is a shorthand property that lets you set all properties for all four of the margin with a single declaration. The following example shows you how to set a margin round a paragraph:

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style="margin: 15px; border:1px solid black;">
```

all four margins will be 15px

```
</p>
```

```
<p style="margin:10px 2%; border:1px solid black;">
```

top and bottom margin will be 10px, left and right margin will be 2% of the total width of the document.

```
</p>
```

```
<p style="margin: 10px 2% -10px; border:1px solid black;">
```

top margin will be 10px, left and right margin will be 2% of the total width of the document, bottom margin will be -10px

```
</p>
```

```
<p style="margin: 10px 2% -10px auto; border:1px solid black;">
```

top margin will be 10px, right margin will be 2% of the total width of the document, bottom margin will be -10px, left margin will be set by the browser

</p>

</body>

</html>

You will see the following result -

all four margins will be 15px
top and bottom margin will be 10px, left and right margin will be 2% of the total width of the document.
top margin will be 10px, left and right margin will be 2% of the total width of the document, bottom margin will be -10px
top margin will be 10px, right margin will be 2% of the total width of the document, bottom margin will be -10px, left margin will be set by the browser

Margin-Bottom

This allows you set up the bottom margin using a value of auto, length or %

<html>

<head>

</head>

<body>

<p style="margin-bottom: 15px; border:1px solid black;">

This is a paragraph with a specified bottom margin.

</p>

<p style="margin-bottom: 5%; border:1px solid black;">

This is another paragraph with a specified bottom margin in percent

</p>

</body>

</html>

The result will be -

This is a paragraph with a specified bottom margin
This is another paragraph with a specified bottom margin in percent

Margin-Top

This allows you to specify the top margin and uses a value of auto, length or %

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style="margin-top: 15px; border:1px solid black;">
```

This is a paragraph with a specified top margin

```
</p>
```

```
<p style="margin-top: 5%; border:1px solid black;">
```

This is another paragraph with a specified top margin in percent

```
</p>
```

```
</body>
```

```
</html>
```

The result will be as follows -

This is a paragraph with a specified top margin

This is another paragraph with a specified top margin in percent

Margin-Left

This property allows you to set the left margin and use a value of %, auto or length

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<p style="margin-left: 15px; border:1px solid black;">
```

This is a paragraph with a specified left margin

```
</p>
```

```
<p style="margin-left: 5%; border:1px solid black;">
```

This is another paragraph with a specified top margin in percent

```
</p>
```

```
</body>
```

```
</html>
```

The result will be as follows -

This is a paragraph with a specified left margin

This is another paragraph with a specified top margin in percent

Margin-Right

This property allows you to specify the right margin, using a value of auto, length or %

```
<html>
  <head>
  </head>
  <body>
    <p style="margin-right: 15px; border:1px solid black;">
      This is a paragraph with a specified right margin
    </p>

    <p style="margin-right: 5%; border:1px solid black;">
      This is another paragraph with a specified right margin in percent
    </p>
  </body>
</html>
```

The result will be shown as follows -

This is a paragraph with a specified right margin

This is another paragraph with a specified right margin in percent

Conclusion

Thank you for purchasing my book, I hope that you found it helpful. As you have seen, CSS is a truly amazing tool for anyone who wants to get into the world of web design. Although it is one of the easiest to understand and learn, I won't promise that you can learn it overnight. Like any computer language, it takes time and it takes patience.

You need to be dedicated to learning it and, once you have, you must be prepared to keep up with all the latest developments. CSS isn't a program you can learn, forget about and then pick up again a few months later. While you will still have the basics in your mind, in those few months, much may have happened and many changes may have been made. It is vital that you keep up with the changes and that you use CSS on a daily basis to ensure that you do not lose any of your newfound skills.

CSS is a program that you learn, that you understand and that you live. Please may I ask a small favor of you - will you leave a review for me at Amazon.com? It isn't only helpful to me; it is helpful to others who may decide to purchase my book.

As stated earlier, this is to reiterate that some of the example code that I have used in this book is attributed to www.tutorialspoint.com