



# **UNIVERSITY OF DAR ES SALAAM**

## **COMPUTING CENTRE**

**Diploma in Computing and Information Technology**



---

**CTT 05105: WEB TECHNOLOGIES**

---

© 2016 University of Dar es Salaam Computing Centre

University Road  
P.O Box 35062  
Dar es Salaam  
Tanzania

Tel: +255 (022) 2410645  
Fax: +255 (022) 2410690  
Email: [training@udsm.ac.tz](mailto:training@udsm.ac.tz)  
Internet: <http://www.ucc.co.tz>

All trademarks acknowledged. E&OE.



© University of Dar es Salaam Computing Centre. No part of this document may be copied without written permission from University of Dar es Salaam Computing Centre unless produced under the terms of a courseware site license agreement with University of Dar es Salaam Computing Centre.

While all reasonable precautions have been taken in the preparation of this document, including both technical and non-technical proofing. University of Dar es Salaam Computing Centre and all staff assume no responsibility for any errors or omissions. No warranties are made, expressed or implied with regard to these notes. University of Dar es Salaam Computing Centre shall not be responsible for any direct, incidental or consequential damages arising from the use of any material contained in this document. If you find any errors in these training modules, please inform University of Dar es Salaam Computing Centre. Whilst every effort is made to eradicate typing or technical mistakes, we apologise for any errors you may detect. University of Dar es Salaam Computing Centre manuals are updated on a regular basis, so your feedback is both valued by us and will help us to maintain the highest possible standards.

<b>Introduction .....</b>	<b>7</b>
COURSE DESCRIPTION .....	7
COURSE OBJECTIVES .....	7
DELIVERY METHODOLOGY .....	7
<b>Chapter 1: BASIC HTML .....</b>	<b>8</b>
1.1 INTRODUCTION TO HTML.....	8
1.1.1 Basic HTML Document.....	8
1.1.2 HTML Document Structure .....	9
1.1.3 Basic HTML Tags.....	10
1.1.4 HTML Text Formatting .....	15
1.1.5 HTML Links .....	17
1.1.6 HTML Tables.....	18
1.1.7 HTML Lists .....	20
1.2 HTML FORMS .....	21
1.2.1 Text Input Controls.....	22
1.2.2 Checkbox Control.....	25
1.2.3 Radio Button Control.....	25
1.2.4 Select Box Control .....	26
1.2.5 File Upload Box.....	27
1.2.6 Button Controls .....	27
1.2.7 Hidden Form Controls.....	28
1.3 HTML IMAGES .....	29
1.4 HTML BACKGROUNDS & COLORS .....	31
1.4.1 Html Background with Colors.....	32
1.4.2 Html Background with Images .....	33
1.4.3 HTML Colors .....	34
EXERCISES .....	36
<b>Chapter 2. ADVANCED HTML .....</b>	<b>37</b>
2.1 THE HTML HEADER ELEMENTS .....	37
2.1.1 The HTML <title> Tag.....	37
2.1.2 The HTML <meta> Tag.....	37
2.1.3 The HTML <base> Tag.....	38
2.1.4 The HTML <style> Tag .....	38
2.1.5 The HTML <script> Tag.....	39
2.1.6 The <noscript> Tag.....	39
2.2. HTML FRAMES .....	40
2.2.1 Disadvantages of Frames .....	40
2.2.2 Creating Frames .....	40
2.2.3 Frame's name and target attributes .....	43
2.3 HTML – IFRAMES .....	44
2.4 HTML 4.0 EVENT ATTRIBUTES.....	46
2.4.1 Window Events .....	46
2.4.2 Form Element Events.....	46
2.4.3 Keyboard Events.....	46
2.4.4 Mouse Events .....	46
EXERCISES .....	47
<b>Chapter 3. BASIC CSS .....</b>	<b>48</b>
3.1 INTRODUCTION TO CSS .....	48
3.1.1 Cascading Order.....	48
3.2 CSS SYNTAX .....	48
3.2.1 The class Selector.....	49
3.2.2 The id Selector.....	50
3.2.3 CSS Comments .....	50
3.3 HOW TO INSERT A STYLE SHEET .....	51
3.3.1 External Style Sheet.....	51
3.3.2 Internal Style Sheet.....	51

3.3.3 <i>Inline Styles</i> .....	52
3.3.4 <i>Multiple Style Sheets</i> .....	52
3.4 CSS BACKGROUND .....	53
3.5 CSS TEXT .....	53
3.5.1 <i>CSS Font Property description</i> .....	53
3.5.2 <i>CSS font-family Property description</i> .....	54
3.5.3 <i>CSS font-style Property description</i> .....	55
3.6 CSS LIST.....	55
3.6.1 <i>CSS List Properties</i> .....	55
3.6.2 <i>CSS list-style-type Property description</i> .....	56
3.7 CSS TABLE PROPERTIES.....	56
3.7.1 <i>CSS border-collapse Property</i> .....	56
3.7.2 <i>CSS border-spacing Property</i> .....	57
3.7.3 <i>CSS table-layout Property</i> .....	57
EXERCISES .....	58
<b>Chapter 4. BASIC JAVASCRIPT .....</b>	<b>59</b>
4.1 INTRODUCTION TO JAVASCRIPT .....	59
4.1.1 <i>Getting started with JavaScript</i> .....	59
4.2 JAVASCRIPT DOCUMENT STRUCTURE.....	60
4.2.1 <i>Where to Put the JavaScript</i> .....	60
4.2.2 <i>Using an External JavaScript</i> .....	61
4.2.3 <i>JavaScript Statements</i> .....	61
4.3 JAVASCRIPT COMMENTS.....	62
4.3.1 <i>JavaScript Multi-Line Comments</i> .....	62
4.3.2 <i>Using Comments to Prevent Execution</i> .....	63
4.3.3 <i>Using Comments at the End of a Line</i> .....	63
4.4 JAVASCRIPT VARIABLES .....	63
4.4.1 <i>Declaring (Creating) JavaScript Variables</i> .....	64
4.5 JAVASCRIPT ARITHMETIC .....	65
4.5.1 <i>JavaScript Arithmetic Operators</i> .....	65
4.5.2 <i>JavaScript Assignment Operators</i> .....	65
4.6 JAVASCRIPT COMPARISON AND LOGICAL OPERATORS .....	66
4.6.1 <i>Comparison Operators</i> .....	66
4.6.1 <i>Logical Operators</i> .....	67
4.6.2 <i>Conditional Operator</i> .....	67
4.7 CONDITIONAL STATEMENTS .....	67
4.7.1 <i>If Statement</i> .....	67
4.7.2 <i>If...else Statement</i> .....	68
4.7.3 <i>If...else if...else Statement</i> .....	69
4.7.4 <i>The JavaScript Switch Statement</i> .....	70
4.8 JAVASCRIPT FUNCTIONS.....	70
4.8.1 <i>How to Define a Function</i> .....	71
4.8.2 <i>JavaScript For Loop</i> .....	72
4.8.3 <i>JavaScript While Loop</i> .....	73
4.8.4 <i>The do...while Loop</i> .....	74
4.8.5 <i>JavaScript break and continue Statements</i> .....	74
4.8.6 <i>JavaScript For...In Statement</i> .....	75
4.9 JAVASCRIPT EVENTS .....	76
4.9.1 <i>onload and onUnload</i> .....	77
4.9.2 <i>onFocus, onBlur and onChange</i> .....	77
4.9.3 <i>onSubmit</i> .....	77
4.9.4 <i>onMouseOver and onMouseOut</i> .....	77
EXERCISES .....	77
<b>Chapter 5. WEB LAYOUT AND DHTML .....</b>	<b>79</b>
5.1 WEB LAYOUTS.....	79
5.1.1 <i>Web Layout - Using Tables</i> .....	79
5.1.2 <i>Multiple Columns Layout - Using Tables</i> .....	80
5.1.3 <i>Web Layouts - Using DIV, SPAN</i> .....	81

5.2 INTRODUCTION TO DHTML .....	82
5.3 DHTML CSS POSITIONING (CSS-P) .....	82
5.3.1 <i>Position</i> .....	83
5.3.2 <i>Visibility</i> .....	83
5.3.3 <i>Filters</i> .....	84
5.3.4 <i>Background</i> .....	85
5.4 DHTML DOCUMENT OBJECT MODEL.....	85
5.5 DHTML EVENT HANDLERS .....	85
EXERCISES .....	86
<b>Chapter 6. BASIC PHP SCRIPTS.....</b>	<b>88</b>
6.1 BASIC PHP SYNTAX .....	88
6.2 COMMENTS IN PHP .....	88
6.3 VARIABLES IN PHP .....	89
6.3.1 <i>Variable Naming Rules</i> .....	89
6.4 STRINGS IN PHP .....	89
6.4.1 <i>The Concatenation Operator</i> .....	90
6.4.2 <i>Using the strlen() function</i> .....	90
6.4.3 <i>Using the strpos() function</i> .....	91
6.5 PHP OPERATORS.....	91
6.6 CONDITIONAL STATEMENTS .....	92
6.6.1 <i>The If...Else Statement</i> .....	92
6.6.2 <i>The Elself Statement</i> .....	93
6.6.3 <i>The Switch Statement</i> .....	93
6.7 ARRAY .....	94
6.7.1 <i>Numeric Arrays</i> .....	95
6.7.2 <i>Associative Arrays</i> .....	95
6.7.3 <i>Multidimensional Arrays</i> .....	96
6.8 LOOPING .....	97
6.8.1 <i>The while Statement</i> .....	97
6.8.2 <i>The do...while Statement</i> .....	97
6.8.3 <i>The for Statement</i> .....	98
6.8.4 <i>The foreach Statement</i> .....	99
6.9 PHP FUNCTIONS .....	99
6.9.1 <i>Create a PHP Function</i> .....	99
6.9.2 <i>Use a PHP Function</i> .....	100
6.9.3 <i>PHP Functions - Adding parameters</i> .....	100
<b>Chapter 7. DEVELOPING INTERACTIVE LAYERS WITH MACROMEDIA.....</b>	<b>102</b>
6.1 UNDERSTANDING LAYERS .....	102
6.2 WORKING WITH LAYERS IN FLASH 8.....	103
6.3 WORKING WITH ADVANCED LAYERS OPTIONS .....	103
6.4 REORGANIZING THE LAYERS WITH FLASH 8.....	105
6.5 LAYER TYPES .....	105
6.6 CREATING FLASH ANIMATION WITH FLASH 8 .....	107
6.6.1 <i>Making the Animation</i> .....	107
6.7 CREATING ROLLOVER IMAGES WITH DREAMWEAVER .....	107
<b>Chapter 8: DRUPAL CORE BASICS .....</b>	<b>111</b>
8.1 DRUPAL INSTALLATION.....	111
8.1.1 <i>Technical requirements for installing Drupal</i> .....	111
8.1.2 <i>Download the Drupal codebase</i> .....	111
8.1.3 <i>Installation through the web interface</i> .....	112
8.2 NODES.....	118
8.2.1 <i>Creating nodes</i> .....	118
8.2.2 <i>Editing nodes and managing revisions</i> .....	121
8.2.3 <i>Other node settings</i> .....	123
8.2.4 <i>View modes for nodes</i> .....	123
8.2.5 <i>Node types and node administration</i> .....	123

8.2.6 Node comments .....	126
8.3 USERS AND PERMISSIONS .....	129
8.3.1 Adding and managing users .....	129
8.3.2 Permissions and roles .....	132
8.3.3 Other user account settings .....	135
8.4 REGIONS AND BLOCKS .....	136
8.4.1 Block settings .....	140
8.4.2 Adding blocks .....	141
8.4.3 Complements and alternatives to blocks .....	142
8.5 MENUS .....	142
8.5.1 Displaying menus .....	142
8.5.2 Creating and editing menu links .....	143
8.5.3 Creating menu links for nodes .....	145
8.6 OTHER BASIC DRUPAL CORE SETTINGS .....	145
8.6.1 Administration aids .....	145
8.6.2 Text formats .....	146
8.6.3 Other settings .....	146
<b>References .....</b>	<b>148</b>

# Introduction

## Course Description

The course introduces students to the concepts and techniques of designing and constructing web pages. It will teach students about basic standards that apply, explore design issues and examine client –side and server- side scripting technologies.

## Course Objectives

At the end of the course students should be able to:

1. Understand the fundamental of the World Wide Web, HTML and web browsers
2. Understand the structure of web documents from a technical perspective
3. Design and construct web pages using techniques such as Cascading Style Sheets (CSS), JavaScript, PHP and DHTML
4. Describe the concepts applied in designing dynamic websites
5. Design dynamic websites using the Concept of DHTML and PHP scripts
6. Design and Manage websites using various HTML editors

## Delivery Methodology

The course will be delivered in form of lecturers, Tutorials in the classroom and in the Computer lab accordingly. Exercise with real life nature will be provided during and at the end of the class. The manual is also designed such that one can follow the course at own time and pace.



## Chapter 1: BASIC HTML

### 1.1 Introduction to HTML

Hyper Text Mark-up Language (HTML) is a vital component in the world of Web page designing and development. The HTML language specifies how a web page should be displayed in a browser.

HTML stands for **Hypertext Markup Language**, and it is the most widely used language to write Web Pages. **Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.

As its name suggests, HTML is a **Markup Language** which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Using HTML tags and elements, you can:

- Control the appearance of the page and the content
- Publish online documents and retrieve online information using the links inserted in the HTML document
- Create online forms, which can be used to collect information about the user, conduct transactions and so on
- Insert objects like audio clips, video clips.

The HTML document forms the source code of a web page. When viewed in the editor, the document is a series of tags and elements that specify how the page is to be displayed. The browser reads the .htm/ .html file and displays the page according to the specified instructions.

The HTML document is displayed in a browser. What is a browser? A browser is an application that reads the HTML source code and displays the page as instructed. To create the source document, an HTML editor is required. Example of HTML editor is Microsoft FrontPage (is a tool that can be used to create, design and edit Web pages. We can also add text, images, tables and other HTML elements to the page. We can also use Notepad to create the HTML document.

In order to view the document in a browser you have to save the document with a .htm / .html extension. HTML commands are called Tags. Tags are used to control the content and appearance of the HTML document. The opening tag is a <>, the closing tag is represented as </>. HTML tags are not case sensitive.

For example, the following HTML syntax will display the message " my first HTML document"

```
<html>
<head>
<title>welcome to HTML</title></head>
<body><h2>my first HTML document</h2></body>
</html>
```

---

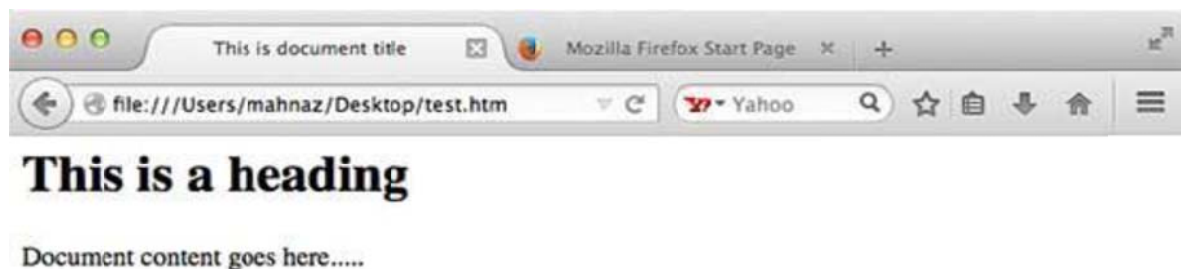
#### 1.1.1 Basic HTML Document

In its simplest form, following is an example of an HTML document:



```
<!DOCTYPE html>
<html>
<head>
<title>This is document title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>Document content goes here.....</p>
</body>
</html>
```

If you are running Windows, start Notepad, type the text above, save it as **test.htm** or **test.html**. Finally open it using a web browser like Internet Explorer or Google Chrome, or Firefox etc. It must show the following output:



As mentioned earlier, HTML is a markup language and makes use of various tags to format the content. These tags are enclosed within angle braces **<Tag Name>**. Except few tags, most of the tags have their corresponding closing tags. For example, **<html>** has its closing tag **</html>** and **<body>** tag has its closing tag **</body>** tag etc.

Above example of HTML document uses the following tags:

Tag	Description
<!DOCTYPE...>	This tag defines the document type and HTML version.
<html>	This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.
<head>	This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.
<title>	The <title> tag is used inside the <head> tag to mention the document title.
<body>	This tag represents the document's body which keeps other HTML tags like <h1>, <div>, <p> etc.
<h1>	This tag represents the heading.
<p>	This tag represents a paragraph.

---

### 1.1.2 HTML Document Structure

A typical HTML document will have the following structure:

```
Document declaration tag
<html>
  <head>
    Document header related tags
  </head>
  <body>
    Document body related tags
  </body>
</html>
```

We have just said that HTML tags are not case sensitive: <B> means the same as <b>. If you want to follow the latest web standards, you should always use lowercase tags. The World Wide Web Consortium (W3C) recommends lowercase tags in their HTML 4 recommendation, and XHTML (the next generation HTML) demands lowercase tags

An HTML document has three basic structures as shown below:

- **The HTML section:** Every HTML document must begin with an opening HTML tag and end with a closing HTML tag <HTML>.....</HTML>.The HTML tags tells the browser that the content between these two tags is an HTML document.
- **The Header Section:** The Header section begins with a <HEAD> tag and is closed with a </HEAD> tag. This section contains the title that is displayed in the navigation bar of the web page. The title itself is enclosed within the TITLE tag, which begins with a <TITLE>tag and is closed with a </TITLE>
- **The BODY section:** This comes after the HEAD section. The BODY section contains the text, images, link that you want to display in your Web page. The BODY section begins with a <BODY> tag and ends with a </BODY>.

---

### 1.1.3 Basic HTML Tags

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

#### Heading Tags

Any document starts with a heading. You can use different sizes for your headings. HTML has six levels of headings, which use the elements <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>. While displaying any heading, browser adds one line before and one line after that heading.

#### Example

```
<!DOCTYPE html>
<html>
<head>
<title>Heading Example</title>
</head>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
</body>
</html>
```

This will produce the following result

**This is heading 1**

**This is heading 2**

**This is heading 3**

### Paragraph Tag

The **<p>** tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening **<p>** and a closing **</p>** tag as shown below in the example:

#### Example

```
<!DOCTYPE html>
<html>
<head>
<title>Paragraph Example</title>
</head>
<body>
<p>Here is a first paragraph of text.</p>
<p>Here is a second paragraph of text.</p>
<p>Here is a third paragraph of text.</p>
</body>
</html>
```

This will produce the following result:

Here is a first paragraph of text.  
Here is a second paragraph of text.  
Here is a third paragraph of text.

### Line Break Tag

Whenever you use the **<br />** element, anything following it starts from the next line. This tag is an example of an **empty** element, where you do not need opening and closing tags, as there is nothing to go in between them.

The **<br />** tag has a space between the characters **br** and the forward slash. If you omit this space, older browsers will have trouble rendering the line break, while if you miss the forward slash character and just use **<br>** it is not valid in XHTML.

#### Example

```
<!DOCTYPE html>
<html>
<head>
<title>Line Break Example</title>
</head>
<body>
<p>Hello<br />
You delivered your assignment on time.<br />
Thanks<br />
Mahnaz</p>
</body>
</html>
```

This will produce the following result:

Hello  
You delivered your assignment on time.  
Thanks  
Mahnaz

### Horizontal Lines

Horizontal lines are used to visually break-up sections of a document. The **<hr>** tag creates a line from the current position in the document to the right margin and breaks the line accordingly. For example, you may want to give a line between two paragraphs as in the given example below:

```
<!DOCTYPE html>
<html>
<head>
<title>Horizontal Line Example</title>
</head>
<body>
<p>This is paragraph one and should be on top</p>
<hr />
<p>This is paragraph two and should be at bottom</p>
</body>
</html>
```

This will produce the following result:

This is paragraph one and should be on top

---

This is paragraph two and should be at bottom

Again **<hr />** tag is an example of the **empty** element, where you do not need opening and closing tags, as there is nothing to go in between them.

The **<hr />** element has a space between the characters **hr** and the forward slash. If you omit this space, older browsers will have trouble rendering the horizontal line, while if you miss the forward slash character and just use **<hr>** it is not valid in XHTML.

### Preserve Formatting

Sometimes, you want your text to follow the exact format of how it is written in the HTML document. In these cases, you can use the preformatted tag **<pre>**.

Any text between the opening **<pre>** tag and the closing **</pre>** tag will preserve the formatting of the source document.

### Example

```
<!DOCTYPE html>
<html>
<head>
<title>Preserve Formatting Example</title>
</head>
<body>
<pre>
```

```
function testFunction( strText ){  
  alert (strText)  
}  
</pre>  
</body>  
</html>
```

This will produce the following result:

```
function testFunction( strText ){  
  alert (strText)  
}
```

Try using the same code without keeping it inside **<pre>...</pre>** tags

### Comments in HTML

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

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

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

### HTML ELEMENTS

An **HTML element** is defined by a starting tag. If the element contains other content, it ends with a closing tag, where the element name is preceded by a forward slash as shown below with few tags:

Start Tag	Content	End Tag
<p>	This is paragraph content.	</p>
<h1>	This is heading content.	</h1>
<div>	This is division content.	</div>

So here **<p>...</p>** is an HTML element, **<h1>...</h1>** is another HTML element. There are some HTML elements which don't need to be closed, such as **<img.../>**, **<hr />** and **<br />** elements. These are known as **void elements**.

HTML documents consist of a tree of these elements and they specify how HTML documents should be built, and what kind of content should be placed in what part of an HTML document.

### HTML Tag vs. Element

An HTML element is defined by a *starting tag*. If the element contains other content, it ends with a *closing tag*.

For example, **<p>** is starting tag of a paragraph and **</p>** is closing tag of the same paragraph but **<p>This is paragraph</p>** is a paragraph element.

### Nested HTML Elements

It is very much allowed to keep one HTML element inside another HTML element:

### Example

```
<!DOCTYPE html>
<html>
<head>
<title>Nested Elements Example</title>
</head>
<body>
<h1>This is <i>italic</i> heading</h1>
<p>This is <u>underlined</u> paragraph</p>
</body>
</html>
```

This will display the following result:

**This is *italic* heading**

This is underlined paragraph

### HTML ATTRIBUTES

We have seen few HTML tags and their usage like heading tags **<h1>**, **<h2>**, paragraph tag **<p>** and other tags. We used them so far in their simplest form, but most of the HTML tags can also have attributes, which are extra bits of information.

An attribute is used to define the characteristics of an HTML element and is placed inside the element's opening tag. All attributes are made up of two parts: a **name** and a **value**:

- The **name** is the property you want to set. For example, the paragraph **<p>** element in the example carries an attribute whose name is **align**, which you can use to indicate the alignment of paragraph on the page.
- The **value** is what you want the value of the property to be set and always put within quotations. The below example shows three possible values of align attribute: **left**, **center** and **right**.

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

### Example

```
<!DOCTYPE html>
<html>
<head>
<title>Align Attribute Example</title>
</head>
<body>
<p align="left">This is left aligned</p>
<p align="center">This is center aligned</p>
<p align="right">This is right aligned</p>
</body>
</html>
```

This will display the following result:

This is left aligned  
This is center aligned  
This is right aligned

<h1> defines the start of a heading.

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

<body> defines the body of an HTML document.

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

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

---

### 1.1.4 HTML Text Formatting

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

#### Text Formatting Tags

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

#### "Computer Output" Tags

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

#### Citations, Quotations, and Definition Tags

Tag	Description
<abbr>	Defines an abbreviation
<acronym>	Defines an acronym
<address>	Defines an address element



<bdo>	Defines the text direction
<blockquote>	Defines a long quotation
<q>	Defines a short quotation
<cite>	Defines a citation
<dfn>	Defines a definition term

## How to View HTML Source

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

## Character Entities

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

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

To display a less than sign in an HTML document we must write: **&lt;** or **&#60;**;

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

**Note** that the entities are case sensitive.

## Non-breaking Space

The most common character entity in HTML is the non-breaking space. Normally HTML will truncate spaces in your text. If you write 10 spaces in your text HTML will remove 9 of them. To add spaces to your text, use the &nbsp; character entity.

## The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	quotation mark	&quot;	&#34;
'	apostrophe	&apos; (does not work in IE)	&#39;

## Some Other Commonly Used Character Entities:

Result	Description	Entity Name	Entity Number
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
€	euro	&euro;	&#8364;
§	section	&sect;	&#167;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
×	multiplication	&times;	&#215;
÷	division	&divide;	&#247;

### 1.1.5 HTML Links

The main power of HTML is the ability to support hyperlinks. A hyperlink, or a link for short, is a connection to another document or file (graphic, audio, video) or even to another section of the same document. We can provide links to:

- A specific section of the same document
- Another document
- Other files-image, audio,
- Another location or server

To create a hyperlink, we need to specify two components

1. The full address or URL of the file to be linked
2. The hotspot that will provide the link. The hotspot may be a line of text or even an image

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

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

The syntax of creating an anchor:

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

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

This anchor defines a link to University Computing Centre:

```
<a href="http://www.ucc.co.tz/">Visit University Computing Centre!</a>
```

#### The Target Attribute

With the target attribute, you can define **where** the linked document will be opened. The line below will open the document in a new browser window:

```
<a href="http://www. ucc.co.tz/"  
target="_blank"> Visit University Computing Centre!</a>
```

#### The Anchor Tag and the Name Attribute

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

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

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

The line below defines a named anchor:

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

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

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

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

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

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

### **Creating an e-mail Link**

If you want users to send an email, you can include a feature within the web page that allows them to send the email from browser. All you have to do is insert the mailto value in the link tag. <a href="mailto:thisperson@mymail.com">

---

## **1.1.6 HTML Tables**

With HTML you can create tables.

### **Tables**

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

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

How it looks in a browser:

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

### **Tables and the Border Attribute**

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

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

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

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

### Headings in a Table

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

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

How it looks in a browser:

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

### Empty Cells in a Table

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

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

How it looks in a browser:

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

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

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

```
<table border="1">
```

```
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>&nbsp;</td>
</tr>
</table>
```

How it looks in a browser:

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

### Table Tags

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

---

### 1.1.7 HTML Lists

Lists are used to group data logically. They can be added to the html document to group related information together. HTML supports ordered, unordered and definition lists.

#### Unordered Lists

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

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

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

Here is how it looks in a browser:

- Coffee
- Milk

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

#### Ordered Lists

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

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

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

Here is how it looks in a browser:

1. Coffee
2. Milk

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

### Definition Lists

A definition list is **not** a list of items. This is a list of terms and explanation of the terms. A definition list starts with the <dl> tag. Each definition-list term starts with the <dt> tag. Each definition-list definition starts with the <dd> tag.

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

Here is how it looks in a browser:

Coffee  
    Black hot drink  
Milk  
    White cold drink

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

## 1.2 HTML FORMS

A form is a section of an HTML document that contains special elements called as controls.

Controls are used to accept input from the user and provide some interaction. The data that is entered by the user can be validated by the client-side scripts and then submitted to the server for further processing.

Uses of forms are collecting names, addresses, telephone numbers, e-mail address and other information to register users for a service or event; gathering information for the purchase of an item; collecting feedback about a Web site and providing a search tool for the web site.

A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

The HTML <form> tag is used to create an HTML form and it has following syntax:

```
<form action="Script URL" method="GET|POST">  
    form elements like input, textarea etc.  
</form>
```

Apart from common attributes, following is a list of the most frequently used form attributes:

#### **Form Attributes**

Attribute	Description
action	Backend script ready to process your passed data.
method	Method to be used to upload data. The most frequently used are GET and POST methods.
target	Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
enctype	You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are: <ul style="list-style-type: none"><li>• application/x-www-form-urlencoded - This is the standard method most forms use in simple scenarios.</li><li>• multipart/form-data - This is used when you want to upload binary data in the form of files like image, word file etc.</li></ul>

#### **HTML Form Controls**

There are different types of form controls that you can use to collect data using HTML form:

- Text Input Controls
- Checkboxes Controls
- Radio Box Controls
- Select Box Controls
- File Select boxes
- Hidden Controls
- Clickable Buttons
- Submit and Reset Button

---

### **1.2.1 Text Input Controls**

There are three types of text input used on forms:

- **Single-line text input controls** - This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.
- **Password input controls** - This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input> tag.
- **Multi-line text input controls** - This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.



### Single-line text input controls

This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML `<input>` tag.

#### Example

Here is a basic example of a single-line text input used to take first name and last name:

```
<!DOCTYPE html>
<html>
<head>
<title>Text Input Control</title>
</head>
<body>
<form >
First name: <input type="text" name="first_name" />
<br>
        Last name: <input type="text" name="last_name" />
</form>
</body>
</html>
```

This will produce the following result:

First name:

Last name:

Following is the list of attributes for `<input>` tag for creating text field.

Attribute	Description
type	Indicates the type of input control and for text input control it will be set to text.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	This can be used to provide an initial value inside the control.
size	Allows to specify the width of the text-input control in terms of characters.
maxlength	Allows to specify the maximum number of characters a user can enter into the text box.

### Password Input controls

This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML `<input>` tag but type attribute is set to **password**.

#### Example

Here is a basic example of a single-line password input used to take user password:

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

```
<head>
<title>Password Input Control</title>
</head>
<body>
<form >
User ID : <input type="text" name="user_id" />
<br>
Password: <input type="password" name="password" />
</form>
</body>
</html>
```

This will produce the following result:

User ID :

Password:

### Multiple-Line Text Input Controls

This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

#### Example

Here is a basic example of a multi-line text input used to take item description:

```
<!DOCTYPE html>
<html>
<head>
<title>Multiple-Line Input Control</title>
</head>
<body>
<form>
Description: <br />
<textarea rows="5" cols="50" name="description">
Enter description here...
</textarea>
</form>
</body>
</html>
```

This will produce the following result:

Description:

Following is the list of attributes for <textarea> tag.

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
rows	Indicates the number of rows of text area box.
cols	Indicates the number of columns of text area box

---

### 1.2.2 Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **checkbox**.

#### Example

Here is an example HTML code for a form with two checkboxes:

```
<!DOCTYPE html>
<html>
<head>
<title>Checkbox Control</title>
</head>
<body>
<form>
<input type="checkbox" name="maths" value="on"> Maths
<input type="checkbox" name="physics" value="on"> Physics
</form>
</body>
</html>
```

This will produce the following result:

☐ Maths   ☐ Physics

Following is the list of attributes for <checkbox> tag.

Attribute	Description
type	Indicates the type of input control and for checkbox input control it will be set to checkbox.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	The value that will be used if the checkbox is selected.
checked	Set to <i>checked</i> if you want to select it by default.

---

### 1.2.3 Radio Button Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **radio**.

#### Example

Here is example HTML code for a form with two radio buttons:

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

```
<head>
<title>Radio Box Control</title>
</head>
<body>
<form>
<input type="radio" name="subject" value="maths"> Maths
<input type="radio" name="subject" value="physics"> Physics
</form>
</body>
</html>
```

This will produce the following result:



The attributes for radio button is similar to those of checkbox

---

### 1.2.4 Select Box Control

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

#### Example

Here is example HTML code for a form with one drop down box

```
<!DOCTYPE html>
<html>
<head>
<title>Select Box Control</title>
</head>
<body>
<form>
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
</form>
</body>
</html>
```

This will produce the following result:



Following is the list of important attributes of <select> tag:

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
size	This can be used to present a scrolling list box.
multiple	If set to "multiple" then allows a user to select multiple items from the menu.

Following is the list of important attributes of <option> tag:

Attribute	Description
-----------	-------------

value	The value that will be used if an option in the selectbox is selected.
selected	Specifies that this option should be the initially selected value when the page loads.
label	An alternative way of labeling options

---

### 1.2.5 File Upload Box

If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to **file**.

#### Example

Here is example HTML code for a form with one file upload box:

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<input type="file" name="fileupload" accept="image/*" />
</form>
</body>
</html>
```

This will produce the following result:



Following is the list of important attributes of file upload box:

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
accept	Specifies the types of files that the server accepts.
accept	Specifies the types of files that the server accepts.

---

### 1.2.6 Button Controls

There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input> tag by setting its type attribute to **button**. The type attribute can take the following values:

Type	Description
submit	This creates a button that automatically submits a form.
reset	This creates a button that automatically resets form controls to their initial values.
button	This creates a button that is used to trigger a client-side script when the user

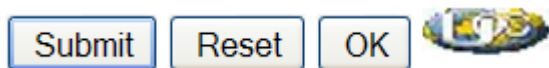
	clicks that button.
image	This creates a clickable button but we can use an image as background of the button.

**Example**

Here is example HTML code for a form with three types of buttons:

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset" value="Reset" />
<input type="button" name="ok" value="OK" />
<input type="image" name="imagebutton" src="ucclogo.png" />
</form>
</body>
</html>
```

This will produce the following result:



---

### 1.2.7 Hidden Form Controls

Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page.

For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page will be displayed next based on the passed current page.

**Example**

Here is example HTML code to show the usage of hidden control:

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<p>This is page 10</p>
<input type="hidden" name="pagename" value="10" />
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset" value="Reset" />
</form>
</body>
</html>
```

This will produce the following result:

This is page 10



## 1.3 HTML Images

Images are very important to beautify as well as to depict many complex concepts in simple way on your web page. This section will take you through simple steps to use images in your web pages.

### Insert Image

You can insert any image in your web page by using **<img>** tag. Following is the simple syntax to use this tag.

```

```

The **<img>** tag is an empty tag, which means that, it can contain only list of attributes and it has no closing tag.

### Example

To try following example, let's keep our HTML file test.htm and image file ucclogo.jpg in the same directory:

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

</body>
</html>
```

This will produce the following result:

Simple Image Insert



You can use PNG, JPEG or GIF image file based on your comfort but make sure you specify correct image file name in **src** attribute. Image name is always case sensitive.

The **alt** attribute is a mandatory attribute which specifies an alternate text for an image, if the image cannot be displayed.



### Set Image Location

Usually we keep all the images in a separate directory. So let's keep HTML file test.htm in our home directory and create a subdirectory **images** inside the home directory where we will keep our image ucclogo.jpg.

#### Example

Assuming our image location is "images/ucclogo.jpg", try the following example:

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

</body>
</html>
```

### Set Image Width/Height

You can set image width and height based on your requirement using **width** and **height** attributes. You can specify width and height of the image in terms of either pixels or percentage of its actual size.

#### Example

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

</body>
</html>
```

This will produce the following result:  
Setting image width and height



### Set Image Border

By default, image will have a border around it, you can specify border thickness in terms of pixels using border attribute. A thickness of 0 means, no border around the picture.

#### Example

```
<!DOCTYPE html>
```

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

</body>
</html>
```

This will produce the following result:  
Setting image Border



### Set Image Alignment

By default, image will align at the left side of the page, but you can use **align** attribute to set it in the center or right.

#### Example

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

</body>
</html>
```

This will produce the following result:  
Setting image Alignment



## 1.4 HTML Backgrounds & Colors

By default, your webpage background is white in color. You may not like it, but no worries. HTML provides you following two good ways to decorate your webpage background.

- Html Background with Colors
- Html Background with Images

Now let's see both the approaches one by one using appropriate examples.

### 1.4.1 Html Background with Colors

The bgcolor attribute is used to control the background of an HTML element, specifically page body and table backgrounds. Following is the syntax to use bgcolor attribute with any HTML tag.

```
<tagname bgcolor="color_value"...>
```

This color\_value can be given in any of the following formats:

```
<!-- Format 1 - Use color name -->
<table bgcolor="lime" >
<!-- Format 2 - Use hex value -->
<table bgcolor="#f1f1f1" >
<!-- Format 3 - Use color value in RGB terms -->
<table bgcolor="rgb(0,0,120)" >
```

#### Example

Here are the examples to set background of an HTML tag:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Background Colors</title>
</head>
<body>
<!-- Format 1 - Use color name -->
<table bgcolor="yellow" width="100%">
<tr><td>
This background is yellow
</td></tr>
</table>
<!-- Format 2 - Use hex value -->
<table bgcolor="#6666FF" width="100%">
<tr><td>
This background is sky blue
</td></tr>
</table>
<!-- Format 3 - Use color value in RGB terms -->
<table bgcolor="rgb(255,0,255)" width="100%">
<tr><td>
This background is green
</td></tr>
</table>
</body>
</html>
```

### 1.4.2 Html Background with Images

The **background** attribute can also be used to control the background of an HTML element, specifically page body and table backgrounds. You can specify an image to set background of your HTML page or table. Following is the syntax to use background attribute with any HTML tag.

**Note:** The *background* attribute is deprecated and it is recommended to use Style Sheet for background setting.

```
<tagname background="Image URL"...>
```

The most frequently used image formats are JPEG, GIF and PNG images.

Here are the examples to set background images of a table.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Background Images</title>
</head>
<body>
<!-- Set table background -->
<table background="/images/html.gif" width="100%" height="100">
<tr><td>
This background is filled up with HTML image.
</td></tr>
</table>
</body>
</html>
```

#### Patterned & Transparent Backgrounds

You might have seen many pattern or transparent backgrounds on various websites. This simply can be achieved by using patterned image or transparent image in the background.

It is suggested that while creating patterns or transparent GIF or PNG images, use the smallest dimensions possible even as small as 1x1 to avoid slow loading.

Here are the examples to set background pattern of a table:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Background Images</title>
</head>
<body>
<!-- Set a table background using pattern -->
<table background="/images/pattern1.gif" width="100%" height="100">
<tr><td>
This background is filled up with a pattern image.
</td></tr>
</table>
<!-- Another example on table background using pattern -->
<table background="/images/pattern2.gif" width="100%" height="100">
```

```
<tr><td>
This background is filled up with a pattern image.
</td></tr>
</table>
</body>
</html>
```

---

### 1.4.3 HTML Colors

Colors are very important to give a good look and feel to your website. You can specify colors on page level using `<body>` tag or you can set colors for individual tags using **bgcolor** attribute.

The `<body>` tag has following attributes which can be used to set different colors:

- **bgcolor** - sets a color for the background of the page.
- **text** - sets a color for the body text.
- **alink** - sets a color for active links or selected links.
- **link** - sets a color for linked text.
- **vlink** - sets a color for *visited links* - that is, for linked text that you have already clicked on.

#### HTML Color Coding Methods

There are following three different methods to set colors in your web page:

- **Color names** - You can specify color names directly like green, blue or red.
- **Hex codes** - A six-digit code representing the amount of red, green, and blue that makes up the color.
- **Color decimal or percentage values** - This value is specified using the `rgb()` property.

Now we will see these coloring schemes one by one.

#### HTML Colors - Color Names

You can specify direct a color name to set text or background color. W3C has listed 16 basic color names that will validate with an HTML validator but there are over 200 different color names supported by major browsers. Here is the list of W3C Standard 16 Colors names and it is recommended to use them.

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

#### Example

Here are the examples to set background of an HTML tag by 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

A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Paintshop Pro or MS Paint.

Each hexadecimal code will be preceded by a pound or hash sign #. Following is a list of few colors using hexadecimal notation.

#### 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 hexa 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

This color value is specified using the **rgb( )** property. This property takes three values, one each for red, green, and blue. The value can be an integer between 0 and 255 or a percentage.

**Note:** All the browsers does not support rgb() property of color so it is recommended not to use it.

Following is a list to show few colors using RGB values.

**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>
```

**Example of colors in different coding**

Color Name	Color HEX	Color RGB
Black	#000000	rgb(0,0,0)
Red	#FF0000	rgb(255,0,0)
Green	#00FF00	rgb(0,255,0)
Blue	#0000FF	rgb(0,0,255)
Yellow	#FFFF00	rgb(255,255,0)
Cyan	#00FFFF	rgb(0,255,255)
Purple	#FF00FF	rgb(255,0,255)
Grey	#C0C0C0	rgb(192,192,192)
White	#FFFFFF	rgb(255,255,255)

## Exercises

1. What is the function of the attribute in HTML tag
2. Provide the difference between HTML tag and HTML element
3. Why is it not safe to use upper case character in HTML tag?
4. Why is not recommended to use RGB values to present colors in the websites?



## Chapter 2. ADVANCED HTML

### 2.1 The HTML Header Elements

The head element contains general information. The <head> tag is a container of various important tags like <title>, <meta>, <link>, <base>, <style>, <script>, and <noscript> tags. The elements inside the head element should not be displayed by a browser.

---

#### 2.1.1 The HTML <title> Tag

The HTML <title> tag is used for specifying the title of the HTML document. Following is an example to give a title to an HTML document:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Title Tag Example</title>
</head>
<body>
<p>Hello, World!</p>
</body>
</html>
```

---

#### 2.1.2 The HTML <meta> Tag

The HTML <meta> tag is used to provide metadata about the HTML document which includes information about page expiry, page author, list of keywords, page description etc.

Following are few of the important usages of <meta> tag inside an HTML document:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Meta Tag Example</title>
<!-- Provide list of keywords -->
<meta name="keywords" content="C, C++, Java, PHP, Perl, Python">
<!-- Provide description of the page -->
<meta name="description" content="Simply Easy Learning by Tutorials Point">
<!-- Author information -->
<meta name="author" content="Tutorials Point">
<!-- Page content type -->
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<!-- Page refreshing delay -->
<meta http-equiv="refresh" content="30">
<!-- Page expiry -->
<meta http-equiv="expires" content="Wed, 21 June 2006 14:25:27 GMT">
<!-- Tag to tell robots not to index the content of a page -->
<meta name="robots" content="noindex, nofollow">
</head>
<body>
<p>Hello, World!</p>
</body>
</html>
```

### 2.1.3 The HTML <base> Tag

The HTML <base> tag is used for specifying the base URL for all relative URLs in a page, which means all the other URLs will be concatenated into base URL while locating for the given item.

For example, all the given pages and images will be searched after prefixing the given URLs with base URL `http://www.tutorial.com/` directory:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Base Tag Example</title>
<base href="http://www.tutorial.com/" />
</head>
<body>

<a href="/html/index.htm" title="HTML Tutorial">HTML Tutorial</a>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<title>HTML link Tag Example</title>
<base href="http://www.tutorial.com/" />
<link rel="stylesheet" type="text/css" href="/css/style.css">
</head>
<body>
<p>Hello, World!</p>
</body>
</html>
```

---

### 2.1.4 The HTML <style> Tag

The HTML <style> tag is used to specify style sheet for the current HTML document. Following is an example to define few style sheet rules inside <style> tag:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML style Tag Example</title>
<base href="http://www.tutorial.com/" />
<style type="text/css">
.myclass{
background-color: #aaa;
padding: 10px;
}
</style>
</head>
<body>
<p class="myclass">Hello, World!</p>
</body>
</html>
```

### 2.1.5 The HTML <script> Tag

The HTML <script> tag is used to include either external script file or to define internal script for the HTML document. Following is an example where we are using JavaScript to define a simple JavaScript function:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML script Tag Example</title>
<base href="http://www.tutorial.com/" />
<script type="text/JavaScript">
function Hello(){
alert("Hello, World");
}
</script>
</head>
<body>
<input type="button" onclick="Hello();" name="ok" value="OK" />
</body>
</html>
```

#### How to Handle Older Browsers

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

---

#### Example

##### JavaScript:

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

##### VBScript:

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

---

### 2.1.6 The <noscript> Tag

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

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

### Example

#### JavaScript:

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

#### VBScript:

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

## 2.2. HTML FRAMES

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

### 2.2.1 Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages:

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
- Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back button* might not work as the user hopes.
- There are still few browsers that do not support frame technology.

### 2.2.2 Creating Frames

To use frames on a page we use `<frameset>` tag instead of `<body>` tag. The `<frameset>` tag defines, how to divide the window into frames. The **rows** attribute of `<frameset>` tag defines horizontal frames and **cols** attribute defines vertical frames. Each frame is indicated by `<frame>` tag and it defines which HTML document shall open into the frame.

#### Example

Following is the example to create three horizontal frames:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Frames</title>
</head>
<frameset rows="10%,80%,10%">
```

```
<frame name="top" src="/html/top_frame.htm" />
<frame name="main" src="/html/main_frame.htm" />
<frame name="bottom" src="/html/bottom_frame.htm" />
<noframes>
<body>
Your browser does not support frames.
</body>
</noframes>
</frameset>
</html>
```

This will produce the following result:

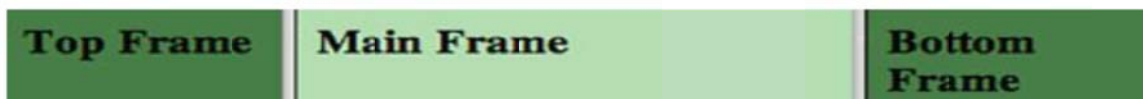


### Example

Let's put the above example as follows, here we replaced rows attribute by cols and changed their width. This will create all the three frames vertically:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Frames</title>
</head>
<frameset cols="25%,50%,25%">
<frame name="left" src="/html/top_frame.htm" />
<frame name="center" src="/html/main_frame.htm" />
<frame name="right" src="/html/bottom_frame.htm" />
<noframes>
<body>
Your browser does not support frames.
</body>
</noframes>
</frameset>
</html>
```

This will produce the following result:



If a user is using any old browser or any browser, which does not support frames then `<noframes>` element should be displayed to the user.

So you must place a `<body>` element inside the `<noframes>` element because the `<frameset>` element is supposed to replace the `<body>` element, but if a browser does not understand `<frameset>` element then it should understand what is inside the `<body>` element which is contained in a `<noframes>` element.

You can put some nice message for your user having old browsers. For example, “*Sorry!! your browser does not support frames*” as shown in the above example.

### **The <frameset> Tag Attributes**

Following are important attributes of the <frameset> tag:

<b>Attribute</b>	<b>Description</b>
cols	<p>Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways:</p> <p>Absolute values in pixels. For example, to create three vertical frames, use <code>cols="100, 500, 100"</code>.</p> <p>A percentage of the browser window. For example, to create three vertical frames, use <code>cols="10%, 80%, 10%"</code>.</p> <p>Using a wildcard symbol. For example, to create three vertical frames, use <code>cols="10%, *, 10%"</code>. In this case wildcard takes remainder of the window.</p> <p>As relative widths of the browser window. For example, to create three vertical frames, use <code>cols="3*, 2*, 1*"</code>. This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.</p>
rows	<p>This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use <code>rows="10%, 90%"</code>. You can specify the height of each row in the same way as explained above for columns.</p>
border	<p>This attribute specifies the width of the border of each frame in pixels. For example, <code>border="5"</code>. A value of zero means no border.</p>
frameborder	<p>This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example <code>frameborder="0"</code> specifies no border.</p>
framespacing	<p>This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example <code>framespacing="10"</code> means there should be 10 pixels spacing between each frames.</p>

### **The <frame> Tag Attributes**

Following are the important attributes of <frame> tag:

<b>Attribute</b>	<b>Description</b>
src	<p>This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, <code>src="/html/top_frame.htm"</code> will load an HTML file available in html directory.</p>
name	<p>This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.</p>

Attribute	Description
frameborder	This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
marginwidth	This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth="10".
marginheight	This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight="10".
noresize	By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize="noresize".
scrolling	This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling="no" means it should not have scroll bars.
longdesc	This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc="framedescription.htm"

---

### 2.2.3 Frame's name and target attributes

One of the most popular uses of frames is to place navigation bars in one frame and then load main pages into a separate frame.

Let's see following example where a test.htm file has following code:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Target Frames</title>
</head>
<frameset cols="200, *">
<frame src="/html/menu.htm" name="menu_page" />
<frame src="/html/main.htm" name="main_page" />
</frameset>
<body>
Your browser does not support frames.
</body>
</frameset>
</html>
```

Here, we have created two columns to fill with two frames. The first frame is 200 pixels wide and will contain the navigation menu bar implemented by **menu.htm** file. The second column fills in remaining space and will contain the main part of the page and it is implemented by **main.htm** file. For all the three links available in menu bar, we have mentioned target frame as **main\_page**, so whenever you click any of the links in menu bar, available link will open in main page.

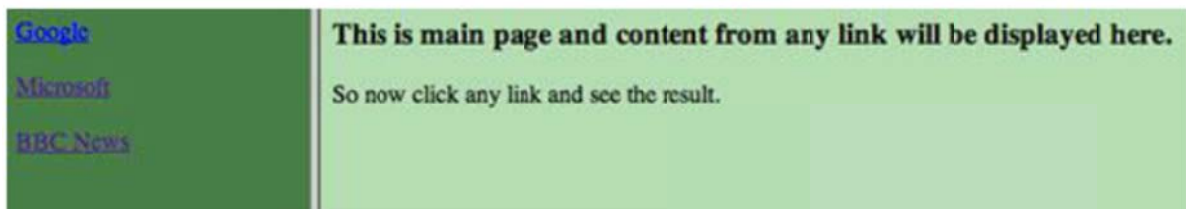
Following is the content of menu.htm file

```
<!DOCTYPE html>
<html>
<body bgcolor="#4a7d49">
<a href="http://www.google.com" target="main_page">Google</a>
<br /><br />
<a href="http://www.microsoft.com" target="main_page">Microsoft</a>
<br /><br />
<a href="http://news.bbc.co.uk" target="main_page">BBC News</a>
</body>
</html>
```

Following is the content of main.htm file:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<h3>This is main page and content from any link will be displayed here.</h3>
<p>So now click any link and see the result.</p>
</body>
</html>
```

When we load **test.htm** file, it produces following result:



Now you can try to click links available in the left panel and see the result. The *target* attribute can also take one of the following values:

Option	Description
_self	Loads the page into the current frame.
_blank	Loads a page into a new browser window.opening a new window.
_parent	Loads the page into the parent window, which in the case of a single frameset is the main browser window.
_top	Loads the page into the browser window, replacing any current frames.
targetframe	Loads the page into a named targetframe.

## 2.3 HTML – IFRAMES

You can define an inline frame with HTML tag **<iframe>**. The **<iframe>** tag is not somehow related to **<frameset>** tag, instead, it can appear anywhere in your document. The **<iframe>** tag defines a rectangular region within the document in which the browser can display a separate document, including scrollbars and borders.

The **src** attribute is used to specify the URL of the document that occupies the inline frame.

### Example

Following is the example to show how to use the **<iframe>**:



```
<!DOCTYPE html>
<html>
<head>
<title>HTML Iframes</title>
</head>
<body>
<p>Document content goes here...</p>
<iframe src="/html/menu.htm" width="555" height="200">
Sorry your browser does not support inline frames.
</iframe>
<p>Document content also go here...</p>
</body>
</html>
```

This will produce the following result:

Document content goes here...

Document content can also go here...

### The <iframe> Tag Attributes

Most of the attributes of the <iframe> tag, including *name*, *class*, *frameborder*, *id*, *longdesc*, *marginheight*, *marginwidth*, *name*, *scrolling*, *style*, and *title* behave exactly like the corresponding attributes for the <frame> tag.

Attribute	Description
src	This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example src="/html/top_frame.htm" will load an HTML file available in html directory.
name	This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
frameborder	This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
marginwidth	This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth="10".
marginheight	This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight="10".
noresize	By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize="noresize".
scrolling	This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling="no" means it should not have scroll bars.
longdesc	This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example

	longdesc="framedescription.htm"
--	---------------------------------

## 2.4 HTML 4.0 Event Attributes

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

---

### 2.4.1 Window Events

Only valid in body and frameset elements

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

---

### 2.4.2 Form Element Events

Only valid in form elements.

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

---

### 2.4.3 Keyboard Events

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

Attribute	Value	Description
onkeydown	<i>script</i>	What to do when key is pressed
onkeypress	<i>script</i>	What to do when key is pressed and released
onkeyup	<i>script</i>	What to do when key is released

---

### 2.4.4 Mouse Events

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

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

## Exercises

1. The <head> tag is a container of various important tags mention them
2. Explain in details the function of HTML <meta> Tag in html document
3. List down the drawbacks of frames
4. Differentiate between frames and iframes
5. List events that are valid only in body and frameset elements

## Chapter 3. BASIC CSS

### 3.1 Introduction to CSS

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

Cascading Style Sheets (CSS) describe how documents are presented on screens. It provides easy and effective alternatives to specify various attributes for the HTML tags. Using CSS, you can specify a number of style properties for a given HTML element. Each property has a name and a value, separated by a colon (:). Each property declaration is separated by a semi-colon (;).

You can use CSS in three ways in your HTML document:

- **External Style Sheet:** Define style sheet rules in a separate .css file and then include that file in your HTML document using HTML <link> tag.
- **Internal Style Sheet:** Define style sheet rules in header section of the HTML document using <style> tag.
- **Inline Style Sheet:** Define style sheet rules directly along-with the HTML elements using **style** attribute.

---

#### 3.1.1 Cascading Order

What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. Browser default
2. External style sheet
3. Internal style sheet (inside the <head> tag)
4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style declared inside the <head> tag, in an external style sheet, or in a browser (a default value).

### 3.2 CSS Syntax

The CSS syntax is made up of three parts: a selector, a property and a value:

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

The selector is normally the HTML element/tag you wish to define, the property is the attribute you wish to change, and each property can take a value. The property and value are separated by a colon, and surrounded by curly braces:

```
body {color: black}
```

**Note:** If the value is multiple words, put quotes around the value:

```
p {font-family: "sans serif"}
```

**Note:** If you wish to specify more than one property, you must separate each property with a semicolon. The example below shows how to define a center aligned paragraph, with a red text color:

```
p {text-align:center;color:red}
```

To make the style definitions more readable, you can describe one property on each line, like this:

```
p
{
text-align: center;
color: black;
font-family: arial
}
```

You can group selectors. Separate each selector with a comma. In the example below we have grouped all the header elements. All header elements will be displayed in green text color:

```
h1,h2,h3,h4,h5,h6
{
color: green
}
```

---

### 3.2.1 The class Selector

With the class selector you can define different styles for the same type of HTML element.

Say that you would like to have two types of paragraphs in your document: one right-aligned paragraph, and one center-aligned paragraph. Here is how you can do it with styles:

```
p.right {text-align: right}
p.center {text-align: center}
```

You have to use the class attribute in your HTML document:

```
<p class="right">
This paragraph will be right-aligned.
</p>
<p class="center">
This paragraph will be center-aligned.
</p>
```

**Note:** To apply more than one class per given element, the syntax is:

```
<p class="center bold">
This is a paragraph.
</p>
```

The paragraph above will be styled by the class "center" AND the class "bold".

You can also omit the tag name in the selector to define a style that will be used by all HTML elements that have a certain class. In the example below, all HTML elements with class="center" will be center-aligned:

```
.center {text-align: center}
```

In the code below both the h1 element and the p element have class="center". This means that both elements will follow the rules in the ".center" selector:

```
<h1 class="center">
This heading will be center-aligned
</h1>
<p class="center">
This paragraph will also be center-aligned.
</p>
```

Do **NOT** start a class name with a number! It will not work in Mozilla/Firefox.

You can also apply styles to HTML elements with particular attributes.

The style rule below will match all input elements that have a type attribute with a value of "text":

```
input[type="text"] {background-color: blue}
```

---

### 3.2.2 The id Selector

You can also define styles for HTML elements with the id selector. The id selector is defined as a #.

The style rule below will match the element that has an id attribute with a value of "green":

```
#green {color: green}
```

The style rule below will match the p element that has an id with a value of "para1":

```
p#para1
{
text-align: center;
color: red
}
```

💡 Do **NOT** start an ID name with a number! It will not work in Mozilla/Firefox.

---

### 3.2.3 CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. A comment will be ignored by browsers. A CSS comment begins with "/\*", and ends with "\*/", like this:

```
/* This is a comment */
p
```

```
{
text-align: center;
/* This is another comment */
color: black;
font-family: arial
}
```

### 3.3 How to Insert a Style Sheet

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

---

#### 3.3.1 External Style Sheet


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

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

The browser will read the style definitions from the file mystyle.css, and format the document according to it.

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension. An example of a style sheet file is shown below:

```
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
```

 Do **NOT** leave spaces between the property value and the units! If you use "margin-left: 20 px" instead of "margin-left: 20px" it will only work properly in IE6 but it will not work in Mozilla/Firefox or Netscape.

---

#### 3.3.2 Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section by using the <style> tag, like this:

```
<head>
<style type="text/css">
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
</style>
</head>
```

The browser will now read the style definitions, and format the document according to it.

**Note:** A browser normally ignores unknown tags. This means that an old browser that does not support styles, will ignore the <style> tag, but the content of the <style> tag will be displayed on the page. It is possible to prevent an old browser from displaying the content by hiding it in the HTML comment element:

```
<head>
<style type="text/css">
<!--
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
-->
</style>
</head>
```

---

### 3.3.3 Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly, such as when a style is to be applied to a single occurrence of an element.

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

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

---

### 3.3.4 Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

For example, an external style sheet has these properties for the h3 selector:

```
h3
{
color: red;
text-align: left;
font-size: 8pt
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3
{
text-align: right;
font-size: 20pt
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:



```
color: red;
text-align: right;
font-size: 20pt
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

## 3.4 CSS Background

The CSS background properties define the background effects of an element. The background property is a shorthand property for setting all background properties in one declaration.

Example

```
body
{
background: #FF0000
}

body
{
background: url(stars.gif) no-repeat top
}

body
{
background: #00FF00 url(stars.gif) no-repeat fixed top
}
```

---

### Possible Values

Value	Description
background-color background-image background-repeat background-attachment background-position	You can declare from one to five background properties in this declaration  Default value: Not defined

## 3.5 CSS Text

The CSS text properties define the appearance of text

---

### 3.5.1 CSS Font Property description

The font property is, with exception of some system fonts, a shorthand property for setting all of the properties for a font in one declaration.

Note: This property also has a sixth value: "line-height", which sets the space between lines.

Example

```
p
{
font: 12px arial
```

```
}  
  
p  
{  
font: italic small-caps bold 12px arial  
}  
  
p  
{  
font: oblique small-caps 900 12px/14px arial  
}  
  
p  
{  
font: menu  
}
```

---

### Possible Values

Value	Description
<i>font-style</i> <i>font-variant</i> <i>font-weight</i> <i>font-size/line-height</i> <i>font-family</i>	Sets the properties for a font. The line-height value sets the space between lines. The value can be a number, a %, or a font size.  Default value: Browser dependent
caption	Defines the font that are used by captioned controls (like buttons, drop-downs, etc.)
icon	Defines the fonts that are used by icon labels
menu	Defines the fonts that are used by dropdown menus
message-box	Defines the fonts that are used by dialog boxes
small-caption	
status-bar	Defines the fonts that are used by window status bars

---

### 3.5.2 CSS font-family Property description

The font-family property is a prioritized list of font family names and/or generic family names for an element. The browser will use the first value it recognizes.

There are two types of font-family values:

- family-name: The name of a font-family, like "times", "courier", "arial", etc.
- generic-family: The name of a generic-family, like "serif", "sans-serif", "cursive", "fantasy", "monospace".

**Note:** Separate each value with a comma, and always offer a generic-family name as the last alternative.

**Note:** If a family-name contains white-space, it should be quoted. Single quotes must be used when using the "style" attribute in HTML.

Example

```
body
{
font-family: courier, serif
}

p
{
font-family: arial, "lucida console", sans-serif
}

<p style="font-family: arial, 'lucida console', sans-serif">
```

---

#### Possible Values

Value	Description
<i>family-name</i> <i>generic-family</i>	A prioritized list of font family names and/or generic family names for an element.  Default value: Browser dependent

---

### 3.5.3 CSS font-style Property description

The font-style property sets the style of a font.

Example

```
body
{
font-style: italic
}
```

---

#### Possible Values

Value	Description
normal	Default. The browser displays a normal font
italic	The browser displays an italic font
oblique	The browser displays an oblique font

## 3.6 CSS List

The CSS list properties allow you to place the list-item marker, change between different list-item markers, or set an image as the list-item marker.

---

### 3.6.1 CSS List Properties

The CSS list properties allow you to place the list-item marker, change between different list-item markers, or set an image as the list-item marker.

**Browser support:** IE: Internet Explorer, F: Firefox, N: Netscape.

The list-style property is a shorthand property for setting all the properties for a list in one declaration.

Example

```
ul
{
list-style: disc outside
}
```

```
}  
  
ol  
{  
list-style: decimal inside  
}
```

---

**Possible Values**

Value	Description
<i>list-style-type</i>	Sets the properties for a list.
<i>list-style-position</i>	
<i>list-style-image</i>	Default value: Not defined

---

### 3.6.2 CSS list-style-type Property description

The list-style-type sets the type of the list-item marker.

Note: Some browsers only support the "disc" value.

Example

```
ul  
{  
list-style-type: disc  
}
```

---

**Possible Values**

Value	Description
none	No marker
disc	Default. The marker is a filled circle
circle	The marker is a circle
square	The marker is a square
decimal	The marker is a number
decimal-leading-zero	The marker is a number padded by initial zeros (01, 02, 03, etc.)
lower-roman	The marker is lower-roman (i, ii, iii, iv, v, etc.)
upper-roman	The marker is upper-roman (I, II, III, IV, V, etc.)
lower-alpha	The marker is lower-alpha (a, b, c, d, e, etc.)
upper-alpha	The marker is upper-alpha (A, B, C, D, E, etc.)
lower-greek	The marker is lower-greek (alpha, beta, gamma, etc.)
lower-latin	The marker is lower-latin (a, b, c, d, e, etc.)
upper-latin	The marker is upper-latin (A, B, C, D, E, etc.)

---

## 3.7 CSS table properties

The CSS table properties allow you to set the layout of a table.

---

### 3.7.1 CSS border-collapse Property

The border-collapse property sets whether the table borders are collapsed into a single border or detached as in standard HTML.

Example

```
table
{
border-collapse: separate
}
```

---

**Possible Values**

Value	Description
separate	Borders are detached
collapse	Default. Borders are collapsed into a single border when possible

---

### 3.7.2 CSS border-spacing Property

The border-spacing property sets the distance between the borders of adjacent cells (only for the "separated borders" model).

Example

```
table
{
border-spacing: 10px
}
```

---

**Possible Values**

Value	Description
<i>length length</i>	Defines the distance in px, cm, etc. If one length parameter is specified, it defines the horizontal and vertical spacing. If two length parameters are specified, the first sets the horizontal spacing and the second sets the vertical spacing. Lengths may not be negative

---

### 3.7.3 CSS table-layout Property

The tableLayout property sets the algorithm used to display the table cells, rows, and columns

Fixed table layout:

- The fixed table layout allows the browser to lay out the table faster than the automatic table layout
- In a fixed table layout, the horizontal layout only depends on the table's width, the width of the columns, and not the content of the cells
- By using fixed table layout, the user agent can begin to display the table once the entire first row has been received

Automatic table layout:

- In an automatic table layout, the column width is set by the widest unbreakable content in the column cells
- This algorithm is sometimes slow since it needs to access all the content in the table before determining the final layout

Example

```
table
{
table-layout: fixed
}
```

---

### Possible Values

Value	Description
automatic	Default. Column width is set by cell content
fixed	Column width is set by table width and the width of the columns

## Exercises

---

1a. What are the three different ways a style sheet can be incorporated into a web page?

b. What would be the advantage of using <link> instead of <style>?

c. What are the different components of this line from a style sheet? What are they called?

p {font-family: courier;}

## Chapter 4. BASIC JAVASCRIPT

### 4.1 Introduction to JavaScript

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.

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

You can keep JavaScript code in a separate file and then include it wherever it's needed, or you can define functionality inside HTML document itself.

---

#### 4.1.1 Getting started with JavaScript

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

---

##### How to Put a JavaScript Into an HTML Page

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

The code above will produce this output on an HTML page:

```
Hello World!
```

---

##### Example Explained

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

So, the `<script type="text/javascript">` and `</script>` tells where the JavaScript starts and ends:

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

The word **document.write** is a standard JavaScript command for writing output to a page.

By entering the `document.write` command between the `<script>` and `</script>` tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page:

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

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

---

### HTML Comments to Handle Simple Browsers

Browsers that do not support JavaScript will display JavaScript as page content.

To prevent them from doing this, and as a part of the JavaScript standard, the HTML comment tag can be used to "hide" the JavaScript. Just add an HTML comment tag `<!--` before the first JavaScript statement, and a `-->` (end of comment) after the last JavaScript statement.

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

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

## 4.2 JavaScript Document Structure

JavaScripts in the body section will be executed WHILE the page loads.  
JavaScripts in the head section will be executed when CALLED.

---

### 4.2.1 Where to Put the JavaScript

JavaScript's in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

**Scripts in the head section:** Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

```
<html>
<head>
<script type="text/javascript">
....
</script>
</head>
```

**Scripts in the body section:** Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

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



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

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

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

---

#### 4.2.2 Using an External JavaScript

Sometimes you might want to run the same JavaScript on several pages, without having to write the same script on every page.

To simplify this, you can write a JavaScript in an external file. Save the external JavaScript file with a .js file extension.

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

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

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

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

---

#### 4.2.3 JavaScript Statements

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

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

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

```
document.write("Hello Dolly");
```

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

The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Because of this you will often see examples without the semicolon at the end.

**Note:** Using semicolons makes it possible to write multiple statements on one line.

---

### JavaScript Code

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

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

---

### JavaScript Blocks

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

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

## 4.3 JavaScript Comments

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

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

---

### 4.3.1 JavaScript Multi-Line Comments

Multi line comments start with /\* and end with \*/.

This example uses a multi line comment to explain the code:

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

---

### 4.3.2 Using Comments to Prevent Execution

In this example the comment is used to prevent the execution of a single code line:

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

In this example the comments is used to prevent the execution of multiple code lines:

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

---

### 4.3.3 Using Comments at the End of a Line

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

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

---

## 4.4 JavaScript Variables

JavaScript variables are used to hold values or expressions.

A variable can have a short name, like **x**, or a more describing name like **length**.

A JavaScript variable can also hold a text value like in **carname="Volvo"**.

Rules for JavaScript variable names:

- Variable names are case sensitive (**y** and **Y** are two different variables)
- Variable names must **begin with a letter** or the underscore character

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

### Example

A variable's value can change during the execution of a script. You can refer to a variable by its name to display or change its value.

---

## 4.4.1 Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the var statement:

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

After the declaration shown above, the variables have no values, but you can assign values to the variables while you declare them:

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

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

---

## Assigning Values to JavaScript Variables

You assign values to JavaScript variables with **assignment statements**:

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

The variable name is on the left side of the = sign, and the value you want to assign to the variable is on the right.

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.

---

## Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that has not yet been declared, the variables will automatically be declared.

These statements:

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

have the same effect as:

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

---

## Redeclaring JavaScript Variables

If you redeclare a JavaScript variable, it will not lose its original value.

```
var x=5;  
var x;
```

After the execution of the statements above, the variable x will still have the value of 5. The value of x is not reset (or cleared) when you redeclare it.

## 4.5 JavaScript Arithmetic

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

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

---

### JavaScript Operators

The operator = is used to assign values.

The operator + is used to add values.

The assignment operator = is used to assign values to JavaScript variables.

The arithmetic operator + is used to add values together.

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

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

---

### 4.5.1 JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

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

---

### 4.5.2 JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that **x=10** and **y=5**, the table below explains the assignment operators:

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

---

### The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
```

```
txt2="nice day";  
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

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

or insert a space into the expression:

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

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

---

### Adding Strings and Numbers

Look at these examples:

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

The rule is:

**If you add a number and a string the result will be a string**

## 4.6 JavaScript Comparison and Logical Operators

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

---

### 4.6.1 Comparison Operators

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

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

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

### How Can it be Used

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

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

---

#### 4.6.1 Logical Operators

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

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

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

---

#### 4.6.2 Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

---

#### Syntax

```
variablename=(condition)?value1:value2
```

---

#### Example

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President" else it will be assigned "Dear".

---

### 4.7 Conditional Statements

Conditional statements in JavaScript are used to perform different actions based on different conditions.

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

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

---

#### 4.7.1 If Statement

You should use the if statement if you want to execute some code only if a specified condition is true.

### Syntax

```
if (condition)
{
  code to be executed if condition is true
}
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

### Example 1

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10
var d=new Date();
var time=d.getHours();
if (time<10)
{
  document.write("<b>Good morning</b>");
}
</script>
```

### Example 2

```
<script type="text/javascript">
//Write "Lunch-time!" if the time is 11
var d=new Date();
var time=d.getHours();

if (time==11)
{
  document.write("<b>Lunch-time!</b>");
}
</script>
```

**Note:** When **comparing** variables you must always use two equals signs next to each other (==)!

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

---

### 4.7.2 If...else Statement

If you want to execute some code if a condition is true and another code if the condition is not true, use the if....else statement.

### Syntax

```
if (condition)
{
  code to be executed if condition is true
}
else
{
  code to be executed if condition is not true
}
```

### Example

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



```
//If the time is less than 10,  
//you will get a "Good morning" greeting.  
//Otherwise you will get a "Good day" greeting.  
var d = new Date();  
var time = d.getHours();  
if (time < 10)  
{  
  document.write("Good morning!");  
}  
else  
{  
  document.write("Good day!");  
}  
</script>
```

---

### 4.7.3 If...else if...else Statement

You should use the if....else if...else statement if you want to select one of many sets of lines to execute.

---

#### Syntax

```
if (condition1)  
{  
  code to be executed if condition1 is true  
}  
else if (condition2)  
{  
  code to be executed if condition2 is true  
}  
else  
{  
  code to be executed if condition1 and  
  condition2 are not true  
}
```

---

#### Example

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

### 4.7.4 The JavaScript Switch Statement

Conditional statements in JavaScript are used to perform different actions based on different conditions.

You should use the switch statement if you want to select one of many blocks of code to be executed.

---

#### Syntax

```
switch(n)
{
case 1:
    execute code block 1
    break;
case 2:
    execute code block 2
    break;
default:
    code to be executed if n is
    different from case 1 and 2
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

---

#### Example

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
    document.write("Finally Friday");
    break;
case 6:
    document.write("Super Saturday");
    break;
case 0:
    document.write("Sleepy Sunday");
    break;
default:
    document.write("I'm looking forward to this weekend!");
}
</script>
```

## 4.8 JavaScript Functions

A function is a reusable code-block that will be executed by an event, or when the function is called.

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to that function.

You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that the function is read/loaded by the browser before it is called, it could be wise to put it in the <head> section.

---

### Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!"
onclick="displaymessage()" >
</form>
</body>
</html>
```

If the line: `alert("Hello world!!")` in the example above had not been put within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before the user hits the button. We have added an `onClick` event to the button that will execute the function `displaymessage()` when the button is clicked.

---

### 4.8.1 How to Define a Function

The syntax for creating a function is:

```
function functionname(var1,var2,...,varX)
{
some code
}
```

`var1`, `var2`, etc are variables or values passed into the function. The `{` and the `}` defines the start and end of the function.

**Note:** A function with no parameters must include the parentheses `()` after the function name:

```
function functionname()
{
some code
}
```

**Note:** Do not forget about the importance of capitals in JavaScript! The word `function` must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

---

### The return Statement

The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

---

### Example

The function below should return the product of two numbers (a and b):

```
function prod(a,b)
{
  x=a*b;
  return x;
}
```

When you call the function above, you must pass along two parameters:

```
product=prod(2,3);
```

The returned value from the prod() function is 6, and it will be stored in the variable called product.

---

### The Lifetime of JavaScript Variables

When you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

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

---

#### 4.8.2 JavaScript For Loop

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

Very often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

---

### The for Loop

The for loop is used when you know in advance how many times the script should run.

#### Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
  code to be executed
}
```

### Example

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 10. i will increase by 1 each time the loop runs.

**Note:** The increment parameter could also be negative, and the <= could be any comparing statement.

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

## Result

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

---

### 4.8.3 JavaScript While Loop

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

---

#### The while loop

The while loop is used when you want the loop to execute and continue executing while the specified condition is true.

```
while (var<=endvalue)
{
    code to be executed
}
```

**Note:** The <= could be any comparing statement.

#### Example

Explanation: The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 10. i will increase by 1 each time the loop runs.

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
{
document.write("The number is " + i);
document.write("<br />");
i=i+1;
}
</script>
</body>
</html>
```

## Result

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

---

#### 4.8.4 The do...while Loop

The do...while loop is a variant of the while loop. This loop will always execute a block of code ONCE, and then it will repeat the loop as long as the specified condition is true. This loop will always be executed at least once, even if the condition is false, because the code is executed before the condition is tested.

```
do
{
    code to be executed
}
while (var<=endvalue);
```

#### Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
{
document.write("The number is " + i);
document.write("<br />");
i=i+1;
}
while (i<0);
</script>
</body>
</html>
```

#### Result

```
The number is 0
```

---

#### 4.8.5 JavaScript break and continue Statements

There are two special statements that can be used inside loops: break and continue.

---

##### Break

The break command will break the loop and continue executing the code that follows after the loop (if any).

##### Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{
if (i==3)
```

```
{  
break;  
}  
document.write("The number is " + i);  
document.write("<br />");  
}  
</script>  
</body>  
</html>
```

### Result

```
The number is 0  
The number is 1  
The number is 2
```

---

### Continue

The continue command will break the current loop and continue with the next value.

### Example

```
<html>  
<body>  
<script type="text/javascript">  
var i=0  
for (i=0;i<=5;i++)  
{  
if (i==3)  
{  
continue;  
}  
document.write("The number is " + i);  
document.write("<br />");  
}  
</script>  
</body>  
</html>
```

### Result

```
The number is 0  
The number is 1  
The number is 2  
The number is 4  
The number is 5
```

---

## 4.8.6 JavaScript For...In Statement

The for...in statement is used to loop (iterate) through the elements of an array or through the properties of an object.

---

### JavaScript For...In Statement

The for...in statement is used to loop (iterate) through the elements of an array or through the properties of an object.

The code in the body of the for ... in loop is executed once for each element/property.

## Syntax

```
for (variable in object)
{
    code to be executed
}
```

The variable argument can be a named variable, an array element, or a property of an object.

---

### Example

Using for...in to loop through an array:

```
<html>
<body>
<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";

for (x in mycars)
{
    document.write(mycars[x] + "<br />");
}
</script>
</body>
</html>
```

## 4.9 JavaScript Events

Events are actions that can be detected by JavaScript.

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger JavaScript functions. For example, we can use the **onClick** event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input box in an HTML form
- Submitting an HTML form
- A keystroke

**Note:** Events are normally used in combination with functions, and the function will not be executed before the event occurs!



---

### 4.9.1 onload and onUnload

The onload and onUnload events are triggered when the user enters or leaves the page.

The onload event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

Both the onload and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

---

### 4.9.2 onFocus, onBlur and onChange

The onFocus, onBlur and onChange events are often used in combination with validation of form fields.

Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30"
id="email" onchange="checkEmail()">
```

---

### 4.9.3 onSubmit

The onSubmit event is used to validate ALL form fields before submitting it.

Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.htm"
onsubmit="return checkForm()">
```

---

### 4.9.4 onMouseOver and onMouseOut

onMouseOver and onMouseOut are often used to create "animated" buttons.

Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver event');return false">

</a>
```

---

## Exercises

1. Examine the following piece of JavaScript code:

```
<script type="text/javascript">
  <!-- // commence script hiding
    document.write("This is text written with JavaScript");
  // end script hiding -->
</script>
```

Answer the following:

- a. where can this code be placed in a document?
- b. the <script> element and its *type* attribute
- c. the comment code present - differentiate between the HTML and JavaScript versions
- d. Why do we include comments?
- e. discuss the <noscript> element - where it is used, and why it is used
- f. the function *write* - what does it do, and how is it called

2. What does the following code yield in a browser?

```
<html>
  <body>
    <script type="text/javascript">
      var i=0
      for (i=0;i<=5;i++)
      {
        if (i!=3)
        {
          continue;
        }
        document.write("The number is " + i);
        document.write("<br />");
      }
    </script>
  </body>
</html>
```

## Chapter 5. WEB LAYOUT AND DHTML

### 5.1 Web Layouts

A webpage layout is very important to give better look to your website. It takes considerable time to design a website's layout with great look and feel.

Now days, all modern websites are using CSS and JavaScript based framework to come up with responsive and dynamic websites but you can create a good layout using simple HTML tables or division tags in combination with other formatting tags. This chapter will give you few examples on how to create a simple but working layout for your webpage using pure HTML and its attributes.

---

#### 5.1.1 Web Layout - Using Tables

The simplest and most popular way of creating layouts is using HTML <table> tag. These tables are arranged in columns and rows, so you can utilize these rows and columns in whatever way you like.

##### Example

For example, the following HTML layout example is achieved using a table with 3 rows and 2 columns but the header and footer column spans both columns using the colspan attribute:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Layout using Tables</title>
</head>
<body>
<table width="100%" border="0">
<tr>
<td colspan="2" bgcolor="#b5dcb3">
<h1>This is Web Page Main title</h1>
</td>
</tr>
<tr valign="top">
<td bgcolor="#aaa" width="50">
<b>Main Menu</b><br />
HTML<br />
PHP<br />
PERL...
</td>
<td bgcolor="#eee" width="100" height="200">
Technical and Managerial Tutorials
</td>
</tr>
<tr>
<td colspan="2" bgcolor="#b5dcb3">
<center>
Copyright © 2014. All Rights Reserved.
</center>
</td>
</tr>
</table>
</body>
```

</html>

This will produce the following result:



---

### 5.1.2 Multiple Columns Layout - Using Tables

You can design your webpage to put your web content in multiple pages. You can keep your content in middle column and you can use left column to use menu and right column can be used to put advertisement or some other stuff.

#### Example

Here is an example to create three column layout:

```
<!DOCTYPE html>
<html>
<head>
<title>Three Column HTML Layout</title>
</head>
<body>
<table width="100%" border="0">
<tr valign="top">
<td bgcolor="#aaa" width="20%">
<b>Main Menu</b><br />
HTML<br />
PHP<br />
PERL...
</td>
<td bgcolor="#b5dcb3" height="200" width="60%">
Technical and Managerial Tutorials
</td>
<td bgcolor="#aaa" width="20%">
<b>Right Menu</b><br />
HTML<br />
PHP<br />
PERL...
</td>
</tr>
</table>
</body>
</html>
```

This will produce the

<b>Main Menu</b>	Technical and Managerial Tutorials	<b>Right Menu</b>
HTML		HTML
PHP		PHP
PERL...		PERL...

---

### 5.1.3 Web Layouts - Using DIV, SPAN

The <div> element is a block level element used for grouping HTML elements while the HTML <span> element is used for grouping elements at an inline level.

Although we can achieve pretty nice layouts with HTML tables, but tables weren't really designed as a layout tool. Tables are more suited to presenting tabular data.

#### Example

Here we will try to achieve same result using <div> tag along with CSS, whatever you have achieved using <table> tag in previous example.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Layouts using DIV, SPAN</title>
</head>
<body>
<div style="width:100%">
<div style="background-color:#b5dcb3; width:100%">
<h1>This is Web Page Main title</h1>
</div>
<div style="background-color:#aaa; height:200px;width:100px;float:left;">
<div><b>Main Menu</b></div>
HTML<br />
PHP<br />
PERL...
</div>
<div style="background-color:#eee; height:200px;width:350px;float:left;">
<p>Technical and Managerial Tutorials</p>
</div>
<div style="background-color:#aaa; height:200px;width:100px;float:right;">
<div><b>Right Menu</b></div>
HTML<br />
PHP<br />
PERL...
</div>
<div style="background-color:#b5dcb3;clear:both">
<center>
Copyright © 2014. All Rights Reserved.
</center>
</div>
</div>
</body>
</html>
```

This will produce the following result:



## 5.2 Introduction to DHTML

DHTML is the art of making HTML pages dynamic!

DHTML is a combination of technologies used to create dynamic and interactive Web sites.

To most people DHTML means a combination of HTML, Style Sheets and JavaScript.

With DHTML a Web developer can control how to display and position HTML elements in a browser window.

---

### The Document Object Model (DOM)

**DOM** stands for the **D**ocument **O**bject **M**odel.

The HTML DOM is the Document Object Model for HTML.

The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML objects.

---

### JavaScript

Allows you to write code to control all HTML elements.

---

#### DHTML Technologies in Netscape 4.x and Internet Explorer 4.x

Netscape 4.x	Cross-Browser DHTML	Internet Explorer 4.x
<ul style="list-style-type: none"><li>• JSS (JavaScript Style Sheets) (allows you to control how different HTML elements will be displayed)</li><li>• Layers (allows you to control element positioning and visibility)</li></ul>	<ul style="list-style-type: none"><li>• CSS1</li><li>• CSS2 (allows you to control how different HTML elements will be displayed)</li><li>• CSS Positioning (allows you to control element positioning and visibility)</li><li>• JavaScript</li></ul>	<ul style="list-style-type: none"><li>• Visual Filters (allow you to apply visual effects to text and graphics)</li><li>• Dynamic CSS (allows you to control element positioning and visibility)</li></ul>

**Note:** Problems with coding DHTML technologies WILL occur as long as each browser creates its own proprietary features and technology that is not supported by other browsers. A Web page may look great in one browser and horrible in another.

## 5.3 DHTML CSS Positioning (CSS-P)

CSS is used to style HTML elements.

**Note:** Most of the DHTML examples require IE 4.0+, Netscape 7+, or Opera 7+!

---

### Which Attributes can be Used with CSS-P?

First, the element must specify the position attribute (relative or absolute).

Then we can set the following CSS-P attributes:

- left - the element's left position
- top - the element's top position
- visibility - specifies whether an element should be visible or hidden
- z-index - the element's stack order
- clip - element clipping
- overflow - how overflow contents are handled

---

#### 5.3.1 Position

The CSS position property allows you to control the positioning of an element in a document.

---

##### position:relative

The position:relative property positions an element relative to its current position.

The following example positions the div element 10 pixels to the right **from where it's normally positioned**:

```
div
{
position:relative;
left:10;
}
```

---

##### position:absolute

The position:absolute property positions an element from the margins of the window.

The following example positions the div element 10 pixels to the right **from the left-margin of its containing block**:

```
div
{
position:absolute;
left:10;
}
```

---

#### 5.3.2 Visibility

The visibility property determines if an element is visible or not.

---

##### visibility:visible

The visibility:visible property makes the element visible.

```
h1
{
visibility:visible;
}
```

### visibility:hidden

The visibility:hidden property makes the element invisible.

```
h1
{
visibility:hidden;
}
```

---

### Z-index

The z-index property is used to place an element "behind" another element. Default z-index is 0. The higher number the higher priority. z-index: -1 has lower priority.

```
h1
{
z-index:1;
}
h2
{
z-index:2;
}
```

In the example above, if the h1 and h2 elements are positioned on top of each other, the h2 element will be positioned on top of the h1 element.

---

### 5.3.3 Filters

The filter property allows you to add more style effects to your text and images.

```
h1
{
width:100%;
filter:glow;
}
```

**Note:** Always specify the width of the element if you want to use the filter property.

The example above produces this output:

---

### Header

---

### Different Filters

**Note:** Some of the Filter properties will not work unless the background-color property is set to transparent!

Property	Argument	Description	Example
alpha	opacity finishopacity style startx starty finishx finisby	Allows you to set the opacity of the element	filter:alpha(opacity=20, finishopacity=100, style=1, startx=0, starty=0, finishx=140, finisby=270)
blur	add direction strength	Makes the element blur	filter:blur(add=true, direction=90, strength=6);
chroma	color	Makes the specified color transparent	filter:chroma(color=#ff0000)
fliph	none	Flips the element horizontally	filter:fliph;
flipv	none	Flips the element vertically	filter:flipv;
glow	color	Makes the element glow	filter:glow(color=#ff0000, strength=5);



	strength		
gray	none	Renders the element in black and white	filter:gray;
invert	none	Renders the element in its reverse color and brightness values	filter:invert;
mask	color	Renders the element with the specified background color, and transparent foreground color	filter:mask(color=#ff0000);
shadow	color direction	Renders the element with a shadow	filter:shadow(color=#ff0000, direction=90);
dropshadow	color offx offy positive	Renders the element with a dropshadow	filter:dropshadow(color=#ff0000, offx=5, offy=5, positive=true);
wave	add freq lightstrength phase strength	Renders the element like a wave	filter:wave(add=true, freq=1, lightstrength=3, phase=0, strength=5)
xray	none	Renders the element in black and white with reverse color and brightness values	filter:xray;

### 5.3.4 Background

The background property allows you to select your own background.

#### background-attachment:scroll

The background scrolls along with the rest of the page.

#### background-attachment:fixed

The background does not move when the rest of the page scrolls.

## 5.4 DHTML Document Object Model

The Document Object Model gives us access to every element in a document.

### How to access an element?

The element must have an id attribute defined and a scripting language is needed. JavaScript is the most browser compatible scripting language, so we use JavaScript.

```
<html>
<body>
<h1 id="header">My header</h1>
<script type="text/javascript">
document.getElementById('header').style.color="red";
</script>
</body>
</html>
```

The script changes the color of the header element.

## 5.5 DHTML Event Handlers

With an event handler you can do something with an element when an event occurs.

With an event handler you can do something with an element when an event occurs: when the user clicks an element, when the page loads, when a form is submitted, etc.

```
<h1 onclick="style.color='red'">Click on this text</h1>
```

The example above defines a header that turns red when a user clicks on it.

You can also add a script in the head section of the page and then call the function from the event handler:

```
<html>
<head>
<script type="text/javascript">
function changecolor()
{
document.getElementById('header').style.color="red";
}
</script>
</head>
<body>
<h1 id="header" onclick="changecolor()">
Click on this text</h1>
</body>
</html>
```

---

## HTML 4.0 Event Handlers

Event	Occurs when...
onabort	a user aborts page loading
onblur	a user leaves an object
onchange	a user changes the value of an object
onclick	a user clicks on an object
ondblclick	a user double-clicks on an object
onfocus	a user makes an object active
onkeydown	a keyboard key is on its way down
onkeypress	a keyboard key is pressed
onkeyup	a keyboard key is released
onload	a page is finished loading. <b>Note:</b> In Netscape this event occurs during the loading of a page!
onmousedown	a user presses a mouse-button
onmousemove	a cursor moves on an object
onmouseover	a cursor moves over an object
onmouseout	a cursor moves off an object
onmouseup	a user releases a mouse-button
onreset	a user resets a form
onselect	a user selects content on a page
onsubmit	a user submits a form
onunload	a user closes a page

## Exercises

---

### 1. Examine the following code and write its output

```
<html>
<head>
<style>
h1.ex1
{
position:relative;
left:20px;
}
h1.ex2
{
```

```
position:relative;
left:-20px;
}
</style>
</head>

<body>
<h1>Normal Heading</h1>
<h1 class="ex1">Heading +20</h1>
<h1 class="ex2">Heading -20</h1>

<p>
Relative positioning moves an element relative to its original position.
</p>
<p>
"left:20" adds 20 pixels to the element's LEFT position.
</p>
<p>
"left:-20" subtracts 20 pixels from the element's LEFT position.
</p>
</body>

</html>
```

---

**2. Describe the event(s) in the following code. Write messages displayed as a result of such event(s)**

```
<html>
<head>

<script type="text/javascript">
function mymessage()
{
alert("This message was triggered from the onunload event");
}
</script>
</head>

<body onunload="mymessage()">

<p>An alert box will display a message when you close this document!</p>
</body>
</html>
```

## Chapter 6. BASIC PHP SCRIPTS

PHP is a powerful server-side scripting language for creating dynamic and interactive websites.

PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP. PHP is perfectly suited for Web development and can be embedded directly into the HTML code.

The PHP syntax is very similar to Perl and C. PHP is often used together with Apache (web server) on various operating systems. It also supports ISAPI and can be used with Microsoft's IIS on Windows

PHP code is executed on the server, and the plain HTML result is sent to the browser.

### 6.1 Basic PHP Syntax

A PHP scripting block always starts with `<?php` and ends with `?>`. A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with `<?` and end with `?>`.

For maximum compatibility, we recommend that you use the standard form (`<?php`) rather than the shorthand form.

```
<?php
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>
<?php
echo "Hello World";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: `echo` and `print`. In the example above we have used the `echo` statement to output the text "Hello World".

Note: The file must have the `.php` extension. If the file has a `.html` extension, the PHP code will not be executed.

### 6.2 Comments in PHP

In PHP, we use `//` to make a single-line comment or `/*` and `*/` to make a large comment block.

```
<html>
<body>
<?php
//This is a comment
/*
This is
a comment
```

```
block
*/
?>
</body>
</html>
```

Variables are used for storing values, such as numbers, strings or function results, so that they can be used many times in a script.

### **6.3 Variables in PHP**

Variables are used for storing a values, like text strings, numbers or arrays.

When a variable is set it can be used over and over again in your script

All variables in PHP start with a \$ sign symbol.

The correct way of setting a variable in PHP:

```
$var_name = value;
```

New PHP programmers often forget the \$ sign at the beginning of the variable. In that case it will not work.

Let's try creating a variable with a string, and a variable with a number:

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

In PHP a variable does not need to be declared before being set.

In the example above, you see that you do not have to tell PHP which data type the variable is.

PHP automatically converts the variable to the correct data type, depending on how they are set.

In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

In PHP the variable is declared automatically when you use it.

---

#### **6.3.1 Variable Naming Rules**

- A variable name must start with a letter or an underscore "\_"
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and \_)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with underscore (\$my\_string), or with capitalization (\$myString)

### **6.4 Strings in PHP**

String variables are used for values that contain character strings.

In this section we are going to look at some of the most common functions and operators used to manipulate strings in PHP.

After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.

Below, the PHP script assigns the string "Hello World" to a string variable called \$txt:

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

The output of the code above will be:

```
Hello World
```

Now, let's try to use some different functions and operators to manipulate our string.

---

### 6.4.1 The Concatenation Operator

There is only one string operator in PHP.

The concatenation operator (.) is used to put two string values together.

To concatenate two variables together, use the dot (.) operator:

```
<?php
$txt1="Hello World";
$txt2="1234";
echo $txt1 . " " . $txt2;
?>
```

The output of the code above will be:

```
Hello World 1234
```

If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string.

Between the two string variables we added a string with a single character, an empty space, to separate the two variables.

---

### 6.4.2 Using the strlen() function

The strlen() function is used to find the length of a string.

Let's find the length of our string "Hello world!":

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

The output of the code above will be:

```
12
```

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

### 6.4.3 Using the strpos() function

The strpos() function is used to search for a string or character within a string.

If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string:

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

The output of the code above will be:

6

As you see the position of the string "world" in our string is position 6. The reason that it is 6, and not 7, is that the first position in the string is 0, and not 1.

## 6.5 PHP Operators

This section lists the different operators used in PHP.

### Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

### Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

### Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true

>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

### Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
	or	x=6 y=3 (x==5    y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

---

## 6.6 Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

- **if...else statement** - use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** - is used with the if...else statement to execute a set of code if **one** of several condition are true

---

### 6.6.1 The If...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

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

Example

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

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



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

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

---

### 6.6.2 The Elself Statement

If you want to execute some code if one of several conditions are true use the elseif statement

Syntax

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

Example

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

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

The Switch statement in PHP is used to perform one of several different actions based on one of several different conditions.

---

### 6.6.3 The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement.

The switch statement is used to avoid long blocks of if..elseif..else code.

### Syntax

```
switch (expression)
{
case label1:
    code to be executed if expression = label1;
    break;
case label2:
    code to be executed if expression = label2;
    break;
default:
    code to be executed
    if expression is different
    from both label1 and label2;
}
```

### Example

This is how it works:

- A single expression (most often a variable) is evaluated once
- The value of the expression is compared with the values for each case in the structure
- If there is a match, the code associated with that case is executed
- After a code is executed, **break** is used to stop the code from running into the next case
- The default statement is used if none of the cases are true

```
<html>
<body>
<?php
switch ($x)
{
case 1:
    echo "Number 1";
    break;
case 2:
    echo "Number 2";
    break;
case 3:
    echo "Number 3";
    break;
default:
    echo "No number between 1 and 3";
}
?>
</body>
</html>
```

An array can store one or more values in a single variable name.

## 6.7 Array

When working with PHP, sooner or later, you might want to create many similar variables. Instead of having many similar variables, you can store the data as elements in an array.

Each element in the array has its own ID so that it can be easily accessed.

There are three different kind of arrays:

- **Numeric array** - An array with a numeric ID key
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

---

### 6.7.1 Numeric Arrays

A numeric array stores each element with a numeric ID key.

There are different ways to create a numeric array.

Example 1

In this example the ID key is automatically assigned:

```
$names = array("Peter","Quagmire","Joe");
```

Example 2

In this example we assign the ID key manually:

```
$names[0] = "Peter";  
$names[1] = "Quagmire";  
$names[2] = "Joe";
```

The ID keys can be used in a script:

```
<?php  
$names[0] = "Peter";  
$names[1] = "Quagmire";  
$names[2] = "Joe";  
echo $names[1] . " and " . $names[2] .  
" are " . $names[0] . "'s neighbors";  
?>
```

The code above will output:

```
Quagmire and Joe are Peter's neighbors
```

---

### 6.7.2 Associative Arrays

An associative array, each ID key is associated with a value.

When storing data about specific named values, a numerical array is not always the best way to do it.

With associative arrays we can use the values as keys and assign values to them.

Example 1

In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Example 2

This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Peter'] = "32";  
$ages['Quagmire'] = "30";  
$ages['Joe'] = "34";
```

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:

```
Peter is 32 years old.
```

---

### 6.7.3 Multidimensional Arrays

In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

Example

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
(
    "Griffin"=>array
    (
        "Peter",
        "Lois",
        "Megan"
    ),
    "Quagmire"=>array
    (
        "Glenn"
    ),
    "Brown"=>array
    (
        "Cleveland",
        "Loretta",
        "Junior"
    )
);
```

The array above would look like this if written to the output:

```
Array
(
    [Griffin] => Array
        (
            [0] => Peter
            [1] => Lois
            [2] => Megan
        )
    [Quagmire] => Array
        (
            [0] => Glenn
        )
    [Brown] => Array
        (
            [0] => Cleveland
            [1] => Loretta
```

```
[2] => Junior
)
)
```

Example 2

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

The code above will output:

```
Is Megan a part of the Griffin family?
```

## 6.8 Looping

Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to perform this.

In PHP we have the following looping statements:

- **while** - loops through a block of code if and as long as a specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as a special condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

---

### 6.8.1 The while Statement

The while statement will execute a block of code if and as long as a condition is true.

Syntax

```
while (condition)
code to be executed;
```

Example

The following example demonstrates a loop that will continue to run as long as the variable i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>
<?php
$i=1;
while($i<=5)
{
    echo "The number is " . $i . "<br />";
    $i++;
}
?>
</body>
</html>
```

---

### 6.8.2 The do...while Statement

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax

```
do
```

```
{  
code to be executed;  
}  
while (condition);
```

**Example**

The following example will increment the value of *i* at least once, and it will continue incrementing the variable *i* as long as it has a value of less than 5:

```
<html>  
<body>  
<?php  
$i=0;  
do  
{  
  $i++;  
  echo "The number is " . $i . "<br />";  
}  
while ($i<5);  
?>  
</body>  
</html>
```

---

**6.8.3 The for Statement**

The for statement is the most advanced of the loops in PHP.

In it's simplest form, the for statement is used when you know how many times you want to execute a statement or a list of statements.

**Syntax**

```
for (init, cond, incr)  
{  
  code to be executed;  
}
```

**Parameters:**

- **init:** Is mostly used to set a counter, but can be any code to be executed once at the beginning of the loop statement.
- **cond:** Is evaluated at beginning of each loop iteration. If the condition evaluates to TRUE, the loop continues and the code executes. If it evaluates to FALSE, the execution of the loop ends.
- **incr:** Is mostly used to increment a counter, but can be any code to be executed at the end of each loop.

**Note:** Each of the parameters can be empty or have multiple expressions separated by commas.

- **cond:** All expressions separated by a comma are evaluated but the result is taken from the last part. This parameter being empty means the loop should be run indefinitely. This is useful when using a conditional break statement inside the loop for ending the loop.

**Example**

The following example prints the text "Hello World!" five times:

```
<html>  
<body>  
<?php
```

```
for ($i=1; $i<=5; $i++)
{
    echo "Hello World!<br />";
}
?>
</body>
</html>
```

---

#### **6.8.4 The foreach Statement**

The foreach statement is used to loop through arrays.

For every loop, the value of the current array element is assigned to \$value (and the array pointer is moved by one) - so on the next loop, you'll be looking at the next element.

Syntax

```
foreach (array as value)
{
    code to be executed;
}
```

Example

The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>
<?php
$arr=array("one", "two", "three");
foreach ($arr as $value)
{
    echo "Value: " . $value . "<br />";
}
?>
</body>
</html>
```

---

### **6.9 PHP Functions**

The real power of PHP comes from its functions.

In PHP - there are more than 700 built-in functions available.

---

#### **6.9.1 Create a PHP Function**

A function is a block of code that can be executed whenever we need it.

Creating PHP functions:

- All functions start with the word "function()"
- Name the function - It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
- Add a "{" - The function code starts after the opening curly brace
- Insert the function code
- Add a "}" - The function is finished by a closing curly brace

Example

A simple function that writes my name when it is called:

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

```
<?php
function writeMyName()
{
    echo "Kai Jim Refsnes";
}
writeMyName();
?>
</body>
</html>
```

---

### 6.9.2 Use a PHP Function

Now we will use the function in a PHP script:

```
<html>
<body>
<?php
function writeMyName()
{
    echo "Kai Jim Refsnes";
}
echo "Hello world!<br />";
echo "My name is ";
writeMyName();
echo "<br />That's right, ";
writeMyName();
echo " is my name.";
?>
</body>
</html>
```

The output of the code above will be:

```
Hello world!
My name is Kai Jim Refsnes.
That's right, Kai Jim Refsnes is my name.
```

---

### 6.9.3 PHP Functions - Adding parameters

Our first function (writeMyName()) is a very simple function. It only writes a static string.

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

You may have noticed the parentheses after the function name, like: writeMyName(). The parameters are specified inside the parentheses.

#### Example 1

The following example will write different first names, but the same last name:

```
<html>
<body>
<?php
function writeMyName($fname)
{
    echo $fname . " Refsnes.<br />";
```



```
}  
echo "My name is ";  
writeMyName("Kai Jim");  
echo "My name is ";  
writeMyName("Hege");  
echo "My name is ";  
writeMyName("Stale");  
?>  
</body>  
</html>
```

The output of the code above will be:

```
My name is Kai Jim Refsnes.  
My name is Hege Refsnes.  
My name is Stale Refsnes.
```

#### Example 2

The following function has two parameters:

```
<html>  
<body>  
<?php  
function writeMyName($fname,$punctuation)  
{  
    echo $fname . " Refsnes" . $punctuation . "<br />";  
}  
echo "My name is ";  
writeMyName("Kai Jim",".");  
echo "My name is ";  
writeMyName("Hege","!");  
echo "My name is ";  
writeMyName("Ståle","...");  
?>  
</body>  
</html>
```

The output of the code above will be:

```
My name is Kai Jim Refsnes.  
My name is Hege Refsnes!  
My name is Ståle Refsnes...
```

## Chapter 7: DEVELOPING INTERACTIVE LAYERS WITH MACROMEDIA

### 6.1 Understanding Layers

Everybody has seen sometimes how the sketchers of cartoons work. And all we've seen how they place a semitransparent leaf with drawings on others and the superposition of all composes the final drawing. Why do they not draw everything on a same leaf? Why do they work with several levels and several drawings if they are going to finish all together?

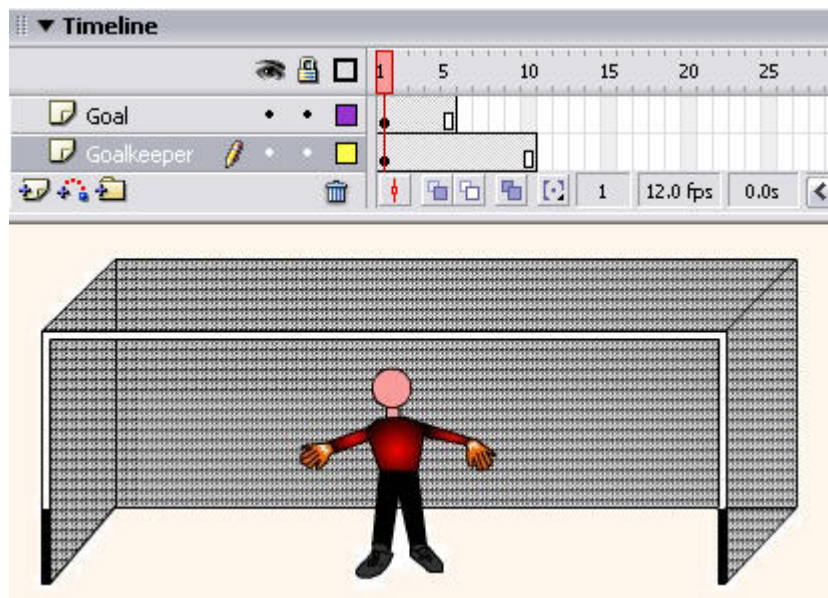
The reasons are many, and these levels that use the sketchers, are equivalent to the **Layers**, which Flash uses. Each layer is, therefore, a level in which we can draw, to insert sounds, texts etc with **INDEPENDENCE** from the rest of layers. It is necessary to take into account that all the layers share the same Timeline and therefore, its different frames will be reproduced simultaneously.

Let's clarify this with an example:

Let's suppose that we have 2 layers. In one layer, the frames from the 1 to the 10 contain the drawing of soccer goal. In the other layer, the frames from the 1 to the 5 contain the drawing of a goalkeeper (they are empty from 5 and further).

Then, this movie will initially display (during the time that lasts the first 5 frames) the goal with the goalkeeper, to show afterward (during the frames from the 5 to the 10) the goal without goalkeeper.

In such a way the goal is independent from the goalkeeper, and we can handle these objects freely, since they do not interfere among themselves.




Another reason to separate the objects in layers is that Flash makes us place each different animation in a layer. Otherwise, all the objects that are in this layer will compose the animation. If we want that an object doesn't form part of an animation, we'll have to delete it from the layer in which this animation is produced.


To proceed further with the example of the goalkeeper, it's easy if we want to create a motion of the goalkeeper towards a side, but if the goal were in the same layer that the goalkeeper, then **BOTH** objects would move towards this side, with which it would be impossible to have only the goalkeeper, moved. The solution is to separate the objects in 2 layers, since we've already done.

In addition, the layers have other utilities, they allow us to order our movie rationally, and they help us in the editing drawings (avoiding that they "are based" on only one, or blocking the rest of layers so that we can only select the layer that is of interest).

## 6.2 Working with Layers in Flash 8


The standard View of a layer is the one that shows the image. Let's see for what the different buttons are used and how to use them.

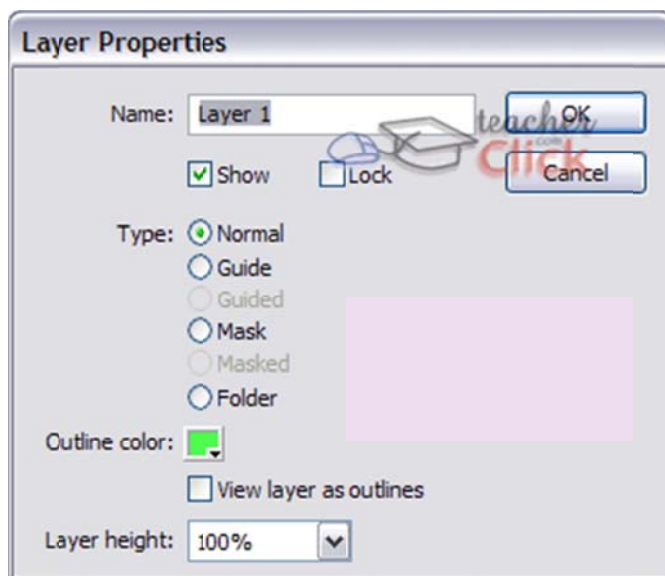
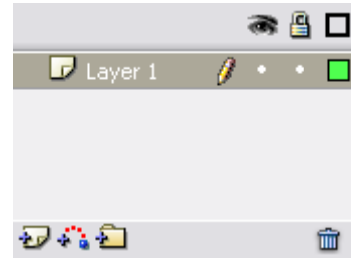
**Insert Layers** : As its name indicates, it is used for **Inserting** layers in the present scene. It inserts normal layers (in the following point the different types from layers will be seen).

**Add Guide Layer** : Insert a kind of guide layer. It is discussed in detail the following point.

**Erase Layer** : **Erase** the selected layer.


**Change Name:** To **change Name:** of a layer, it is enough to double click the current name.


**Layer Properties:** If we double click the icon , we'll be able to access a panel with the **properties** of the layer we've clicked. We'll be able to modify all the options that we've previously commented and some more of lesser importance.




Here you can change different options about the layer, like its name or color. You can also lock or hide it.

## 6.3 Working with Advanced Layers Options

**Show /Hide Layers** : This button allows us **to show and hide** all layers of the movie. It is very helpful when we have many layers and we want only to see one of them. In order to activate the view of a concrete layer (or to hide it) it is enough to click the corresponding layer in the point (or in the cross) that is under the icon "Show/Hide layers"

**Block Layers** : **It blocks** the edition of all the layers, so we'll not be able to edit them until unblocking them. In order to block or to unblock a concrete layer, we'll proceed like in the previous point, clicking on the point or icon "Lock" located in the current layer under the icon "Block Layers".

To block a layer is very useful when we have several objects together in different layers and want to make sure that we don't modify "without wanting" some of them. After blocking a layer we'll be able to work with the security of not modifying its objects, not even to select them, so we'll edit the desired object with greater easiness.

**Show/Hide layers as outlines** : This button show / hide the contents of all the layers as if they were composed only by **borders**. Facing numerous set of objects in this way, we'll be able to distinguish all of them easily and to see in what layer each of them is.

Every layer can be also activated or deactivated by using the above mentioned button in a similar manner.

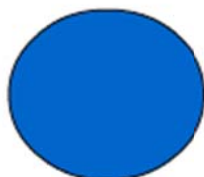
Let's see how these activated and deactivated options are shown.



In the first image the current layer doesn't have any of the buttons activated, we can observe that a black point appears in the column "Show Layers". This point means that this option isn't activated; the same happens to the button "Block layers". There are a FILLED square in the column "Show layers as outlines", which symbolizes that the objects will be completed and not only its outlines.

There appears a cross located under the column "Show Layers" in the second image, which indicates that this layer isn't visible on the stage. It appears a lock under the column "block layers", which symbolizes that the layer is blocked. And the fill does NOT appear in the column "Show layers as outlines". The layer is shown in this way and we'll not be able to see the fillings until we deactivate this option.

In addition, the color of the outlines will be different for each layer, so that we can distinguish them better. The color of the outline will coincide with the color indicated in each layer. In this example you can see the object depending on the outline option activated or not:



## 6.4 Reorganizing the Layers with Flash 8

We've dedicated a whole unit to how to place objects, we already know how to get an object over another one in a movie, how to group them and many other things. But we'll have realized that if we work with different layers, all this isn't very helpful.

As it has been already commented, the different layers have many features in common with others. The first and main feature is the Timeline, all the layers of a same scene share the same timeline and therefore, the objects of all frames from all the layers will be seen at the same time in the movie as superposed ones upon others.

But what object is over the others? The criterion is given by the Layers position in the movie. The objects that will appear ahead of all the others will be those that are on the top layer.

Pay attention to the previous example:

The goalkeeper appears in front of the goal, because the layer "Goalkeeper" is located upon the layer "Goal", so it can be seen in the image. If we want to change this distribution, it is enough to click the layer that we want to move and drag it upwards or downwards or to the desired position.


We'll see how the objects are placed ahead or behind those of the selected layer according to whether its layer is above or below this one.




In order to move a frame from a layer to another one, it is enough to select the frame to move and drag it up to the layer where we want to stick it. The frame can also be Copied and then pasted in the destiny layer.

## 6.5 Layer Types

There are many types of layers, as you can see in the general properties of a layer:

**Normal Layers** : These are the layers of Flash by default, and they have all the properties described in the previous points. They are used most commonly and for almost everything: to place objects, sounds, actions, and others

**Guide Layers** : These are layers for special or specific content. They are used in the animations of objects movement and its only aim is to set the trajectory that this object must follow. Because its mission is to represent the trajectory of an animated object, its content usually is a line (straight, curve, etc).

In this image we can see the content of 2 layers. The first of them contains the blue ball and the second contains the curved line. We have defined the second layer as Guide Layer, so that when making the movement animation (this we'll see it in a next unit) it will be used as a track for the blue ball. Its content wouldn't be seen in the movie.



It is important to remember that the content of the Guide Layers will not be seen in the final movie. Its effect will cause that the blue ball moves from one end of the line to the other following that way. That's a beautiful effect.

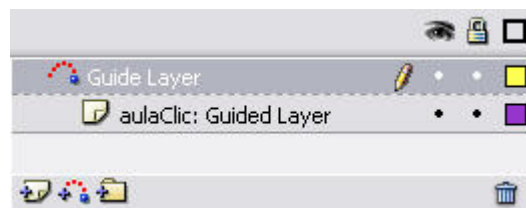
**Guided Layers** : When we define a layer as a guide layer, it is necessary to define also a guided layer. This is a layer that will be affected by the guide defined in the guide layer.

If we didn't define a guided layer, the guide layer will have not any effect and though it will not be seen in the movie (being a guide layer) it will not cause any effect in the other layers. In the previous image, the blue ball might have to be found in a Guided layer; otherwise it'll not follow the way set by the guide layer.


The guide layers and the guided layers are related to each other in an evident way. A series of guided layers correspond to every guide layer.


On associating a guide layer with a guided layer, a change on the guide layer icon will indicate that the job correctly done.

In the image we can see an example of a guide layer and guided layers associated correctly.



The use of the Guide Layers and its utilities we'll see in detail in the unit of Motion Animations

**Mask Layers** : These layers can be seen as groups that keep the unmasked layers off. The use of these layers is something that needs high concentration. It is enough to mention that these layers are placed "above" the layers, which they mask, and allow us to see only the part of the layer that cover up the objects located in the mask layer (they act like filters). Similar to the guide layers, the existing objects in this type of layers are not seen either in the final movie. Only the objects from their masked layer corresponding to the "covered" ones we'll be seen.

**Masked Layers** : These layers work jointly with the masked Layers. The mask layers and the masked layers must be associated to be involved correctly.

Its objects are visible in the final movie, but only when some object of the Mask layer is on the top of them.



Let's see the operation of these layers on an example.

In this example, the blue rectangles are part of a Masked Layer and therefore they will be seen in the final movie (but only ones covered by the mask layer). The red oval is located in the Mask layer and it will not be seen in the movie, but only what "covers" it will be seen. Thus the masks are displayed in the following way.





## 6.6 Creating Flash animation with Flash 8

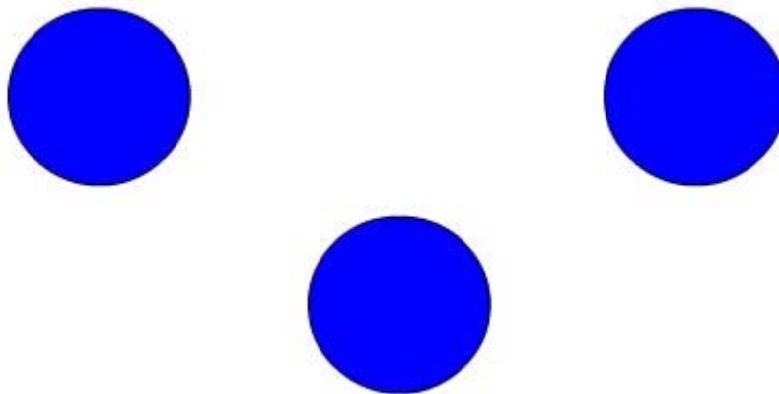
One of the main characteristics of Flash 8 is its simplicity; the straightforwardness in its use allows creating animations in an effective and quick way.

Let's suppose you want to create an animation in which a globe goes up and down. It may seem a task that will take long hours, but not really. Let's see how easy is to handle it with Flash.

---

### 6.6.1 Making the Animation

At first glance, it seems logical to draw the globe at each moment, so that growing number of moments makes the movement more real: the more drawn instants, the more realistic movement. Nevertheless, with Flash it is sufficient to create only 3 frames: firstly we will draw the globe at the initial instance (above all), secondly, we will draw the globe at the moment when it touches the ground and then the globe will come back to its initial position (actually you can create this frame by making a copy of the first one). So far as we see now, most part of the work (drawing objects) is already done.



Now, the duration of each movement is determined by setting the time between the moments when the globe is at the top and at the bottom, and finally Flash is pointed out to create an animation of movement between those two frames.

## 6.7 Creating rollover images with Dreamweaver

Layers are a special kind of HTML elements, which can be used as a container to hold other HTML elements and to show them dynamically using JavaScript. We can stack more than one layer over another. One or more Layers can be made visible dynamically, by hiding

others. But using Macromedia Dreamweaver you can do all this without even knowing JavaScript or coding.

The disadvantage with layers is, they can be viewed only with 4.0 and above browsers. In this section we are going to use multiple layers stacked over another, and make it viewable, when the mouse is rolled over in an Image. In this case study I have taken 2 rollover images Image1, Image 2 and three layers: Layer1, Layer2 and Layer3.

When we mouse over on the Image1, the second layer will appear while Layer1 & Layer 3 will be hidden along with its content. Likewise for the Image 3, Layer1 and 2 will be hidden and only Layer3 will appear with its content. When we mouse out from the Images Layer 1 will come up again. To achieve this Follow the steps and enjoy!

---

### 6.7.1 Creating rollover Images:

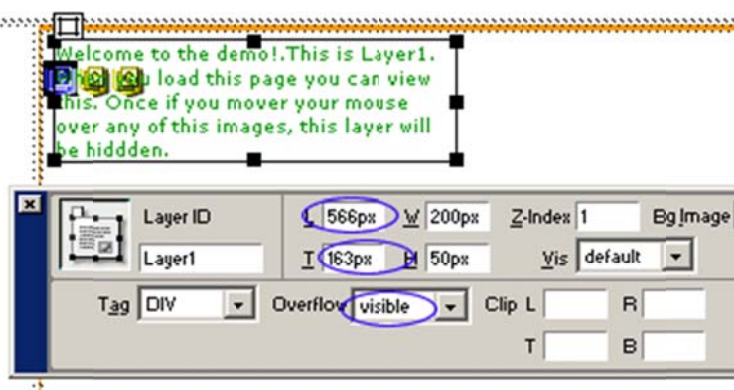
First let us create 2 Images, we can use Macromedia Fireworks to create the images, in our case I have designed 4 buttons; 2 for the up state and 2 for the down state. After designing the buttons, open a new page in Dreamweaver, create a 2X1 table. Now Insert the images using, Insert>Interactive Images>Rollover Image. Browse and select the first one for the up state and the second one for the down state. Now do it for the next Image, so that both the Images are in the first column of the table as shown in this example. Now it should look like this.



---

### 6.7.2 Placing Layers

Open Insert>Layer and place the Layer1. Move the layer into the column2 of the table. Now you can write the content inside the Layer. Select the layer in the Properties Inspector and select Overflow as 'visible'. This makes this first layer as default layer and makes visible while the page is loaded.

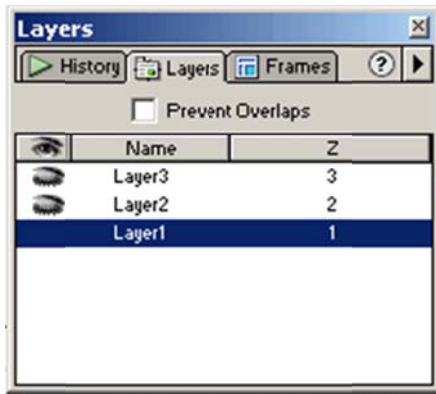


Note down the position of the Layer (566px and 163px). Insert the second and third layers on the same position, so that they will sit over the first layer. Now insert the text on those layers. (Note: Due to the over lapping of layers the text may not be clear. So, you can place the layers at somewhere else, write down the text and then change the position of the layers to sit on the first layers position)

---


### 6.7.3 Changing the Layer properties



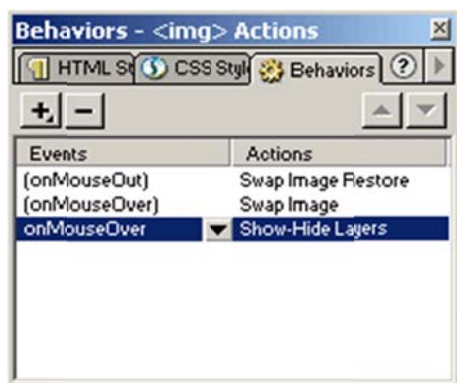
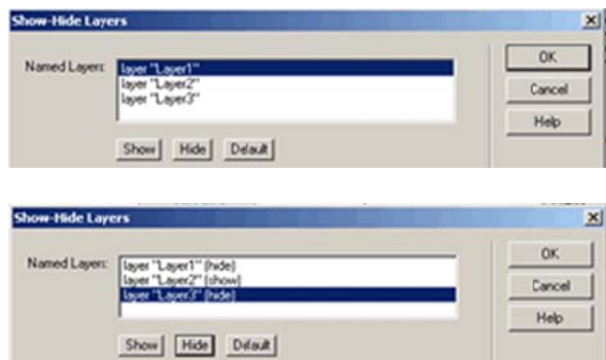


Select Window>Layers. You will get the Layer Inspector, with all the three layer details. Select the Layer2 and Layer3 and If the eye is open click on it to close for both of them. In the properties Inspector you can see the Overflow is selected as hidden for Layer2 and Layer3. This enables the Layer2 and 3 to be hidden while the page is loaded.

### 6.7.4 Adding Behaviors

Now select the Image1 where we are going to add the dynamic Layer effects. Then Open Window > Behavior. You can see the Events and actions created by Dreamweaver for the Rollover Images which we did in the step1. Click on the  button and select "Show-hide Layers". You will get this box to 'Show or Hide' the Layers.

Now do the following;  
a. Select Layer1 then click on 'Hide'  
b. Select Layer2 and click on 'Show' then,  
c. Select Layer3 and click on 'Hide'.  
This action will make the Layer1 and 3 to disappear and to show the Layer2 while you move your mouse over the Image1.




You can also see the new JavaScript events created by Dreamweaver in the Behaviors Inspector. Here the new onMouseOver and onMouseOut with Show-hide Layers as actions.

Note here, You can use other events also as you like. For example instead of onMouseOver you can use onClick. To do this you need to click on the tiny black, inverted arrow next to the event. Use this to select other events, ie OnClick, Down etc.,. At the foot of the list there is a category which allows you to choose which browsers to support, make sure a check is next to 4 and higher browsers. (Only after choosing the browser 4 or higher you will be able to see all the Events in a list)

Follow the step 4 by selecting Image2. Then, **a.** Select Layer1 then click on 'Hide'; **b.** Select Layer2 and click on 'Hide' then, **c.** Select Layer3 and click on 'Show'. This action will make the Layer1 and Layer2 to disappear and to show the Layer2 while you move your mouse over the Image1.

### 6.7.5 Bringing default layer again:

Now as a final step, you need to bring back the Layer1 when you move your mouse out of the Image (Or when you mouse away from the images at somewhere on that page). Select the Image1 and click on the  button and select "Show-hide Layers". You will get this box to 'Show or Hide' the Layers.

Now do the following;

- Select Layer1 than click on 'Show'
- Select Layer2 and click on 'Hide' then,
- Select Layer3 and click on 'Hide'.



This action will make the Layer2 and Layer3 to disappear and to show the Layer1 while you move your mouse out of the Image1. Also do not forget to change the Events to **onMouseOut**.

Repeat step 5 with Image2 also. Now you should get the Behavior Inspector like that Image, showing all the Events for both Image1 & Image2.

## Chapter 8: DRUPAL CORE BASICS

Drupal is a program, or a web application, used to manage content on a website. It is distributed with a license usually called *open source*.

### 8.1 Drupal Installation

#### 8.1.1 Technical requirements for installing Drupal

Drupal is written in PHP and uses a database to store much of the information being handled on your site. The most common platform for running Drupal is a so-called LAMP stack – Linux, Apache, MySQL, PHP – but Drupal will run on any platform that can run PHP and has a database usable by Drupal. You can, for example, run Drupal on Windows or Mac platforms.

Drupal 7 runs all database queries through an abstraction layer, PHP Data Objects (PDO), which theoretically allows Drupal to run on a wide variety of databases. To work in practice, though, you need drivers to manage the interpretation necessary between Drupal and PDO. This limits your options of databases to use, but the most common databases already have drivers ready to use with Drupal. Drupal supports MySQL (and compatible databases, such as MySQLi and MariaDB), PostgreSQL, SQLite, MS SQL, Oracle databases and the non-relational database MongoDB.

A clean install of Drupal 7 requires PHP to use about 32 MB of memory, but a full website will most likely require much more. The exact memory requirements depends on how a site is set up (and optimized), but a Drupal developer with the *php memory limit* setting at 128 MB will rarely need to care about the site's memory.

You install Drupal by the following steps:

1. Download Drupal and put it on a server.
2. Make two adjustments to Drupal's file system (or make sure that Drupal can do them for you).
3. Provide Drupal with the login information for your database.
4. Set a few basic settings for your site, such as information for the first user account.

These steps are described in more detail below.

#### 8.1.2 Download the Drupal codebase

You will find the latest version of Drupal at [drupal.org](http://drupal.org). There is a big green button *get started with Drupal*, providing you with some easy-to-use links to get started. There is also a link directly to *Drupal core* on the site's front page.

The downloaded file package is either a zip or a tar.gz archive, depending on which download link you use. Regardless of its format, the archive should be extracted with a suitable tool and its content should be moved to the web folder on your server.

Drupal won't care if you place all the files in a sub folder of your server's web folder or directly in the web root. The only thing affected is the path used for accessing your site.

If you're using a web hosting service, you will probably need an *ftp client* to upload files. A good and free ftp client is FileZilla. You will need ftp login information, provided by your web hosting service.

Before the actual installation can start, you will (on most servers) need to make two adjustments to the file system. This ensures Drupal is able to write files to two places. The required steps are:

1. In Drupal's root folder there is a subfolder *sites*, containing a subfolder *default*. Within it is a file named *default.settings.php*. You should copy this and give the resulting file the name *settings.php* (placed in the same folder). Note that the file should be copied, and not renamed. Drupal requires the original file as well.
2. In the same folder (*sites/default*), you should create a subfolder named *files*.
3. The file and folder you have created should both be writable for Drupal, meaning that they should be writable for your server. These are the only places Drupal requires write access (except for a temporary folder that should be outside the web folder and usually are inherited from your server settings without you having to even think about it).

If you're using an ftp client, you can change the write access for files and folders by right-clicking on them and choosing any option similar to *properties* or *file attributes*. How the settings appear depends on your ftp client, but you should be able to find a setting for *group write access*. Check that option.

Depending on how your server is set up, Drupal may be able to make these changes itself during the installation process. If you find this step confusing, skip it and try installing Drupal anyway. Drupal will let you know if some requirements are unmet, and provide suggestion to how to get things working.

---

### 8.1.3 Installation through the web interface

When the file structure is in place, you do the actual Drupal installation by visiting your soon-to-be website. The server will find Drupal's *index.php* file and if no database environment is available, Drupal will run the installation wizard. The steps in the wizard are:

1. Select an installation profile. Drupal core contains the *Standard* and *Minimal* profiles. (See figure A1.2) The *standard* profile contains a number of commonly used settings, while the *minimal* profile is as bare as it gets. There are a number of other installation profiles available at [drupal.org](http://drupal.org), for example for building news sites or project management sites. The examples in this manual all build on the standard profile.

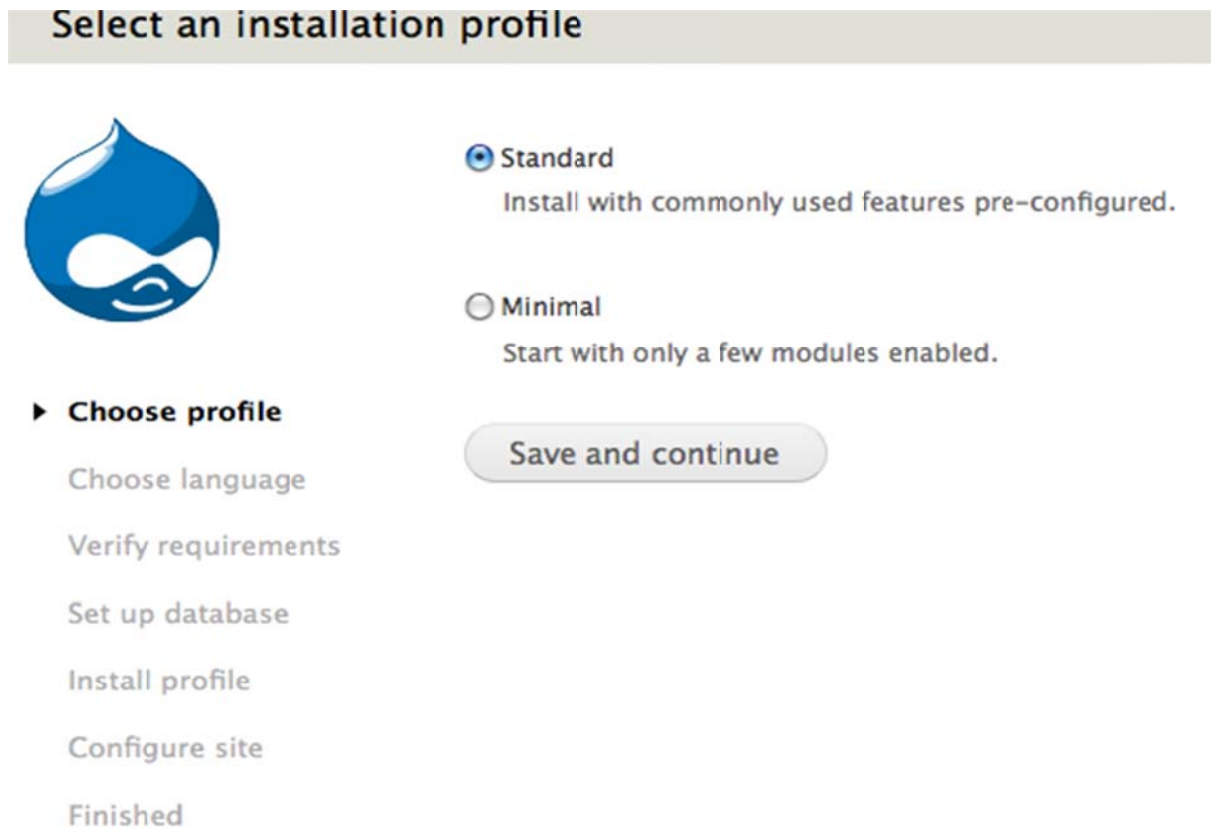


Figure A1.2: Professional Drupal developers tend to like the minimal installation profile. Others tend to like the standard profile.


2. Choose language. By default only English is available, but if you (or the installation profile) have downloaded additional language packs, Drupal will recognize them and offer alternatives on this screen. (See figure A1.3) You can add new languages while building your site, as well as during the installation.



Figure A1.3: By adding more languages you can use Drupal in non-English languages even during the installation.

3. When language settings are complete, Drupal will check that all requirements are met, such as the file and folder described in the previous section are writable. If anything is wrong, and Drupal can't fix it by itself, you will get an error message along with notes about how to fix the problem.
4. Database configuration. This includes selecting the type of database and entering information necessary to access the database, database name, username and any password. (See figure A1.4) If you use a web hosting service only providing you with a single database, you may separate different installations by using a *table prefix*. This is short text added before all database tables created by Drupal (such as *site1\_* or *site2\_*).

## Database configuration



✓ Choose profile

✓ Choose language

✓ Verify requirements

▶ Set up database

Install profile

Configure site

Finished

**Database type \***

☒ MySQL, MariaDB, or equivalent

☐ PostgreSQL

☐ SQLite

The type of database your Drupal data will be stored in.

**Database name \***

The name of the database your Drupal data will be stored in. It must exist on your server before Drupal can be installed.

**Database username \***

**Database password**

**ADVANCED OPTIONS**

These options are only necessary for some sites. If you're not sure what you should enter here, leave the default settings or check with your hosting provider.

**Database host \***

If your database is located on a different server, change this.

**Database port**

If your database server is listening to a non-standard port, enter its number.

**Table prefix**

If more than one application will be sharing this database, enter a table prefix such as *drupal\_* for your Drupal site here.


Save and continue

Figure A1.4: Drupal needs database login information to work properly.

5. When the database settings have been entered, Drupal will run the actual installation. This means that Drupal will be busy for a minute or so and displaying the different steps in the installation process as entertainment.
6. Configure site. When the installation is complete, you will be asked for basic site information, such as site name and time zone settings. (See figure A1.5) You should also enter information for the first user account on the site. This account will be granted all permissions on the site and should only be used during development and system updates. (It should not be used as a personal account used to post content to the site.)



## Configure site



- ✓ Choose profile
- ✓ Choose language
- ✓ Verify requirements
- ✓ Set up database
- ✓ Install profile
- **Configure site**

Finished

### SITE INFORMATION

**Site name \***

**Site e-mail address \***  
  
Automated e-mails, such as registration information, will be sent from this address. Use an address ending in your site's domain to help prevent these e-mails from being flagged as spam.

### SITE MAINTENANCE ACCOUNT

**Username \***  
  
Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

**E-mail address \***

**Password \***  
 Password strength: **Strong**

**Confirm password \***  
 Passwords match: yes

To make your password stronger:  
• Add lowercase letters

### SERVER SETTINGS

**Default country**  
  
Select the default country for the site.

**Default time zone**  
  
By default, dates in this site will be displayed in the chosen time zone.

### UPDATE NOTIFICATIONS

☒ Check for updates automatically

☒ Receive e-mail notifications

The system will notify you when updates and important security releases are available for installed components. Anonymous information about your site is sent to [Drupal.org](http://Drupal.org).

Save and continue

Figure A1.5: The last step is entering some basic site information, including information for user account 1.

7. When all these settings are made, your installation is complete. (See figure A1.6) You can now start the fun of experimenting with Drupal!

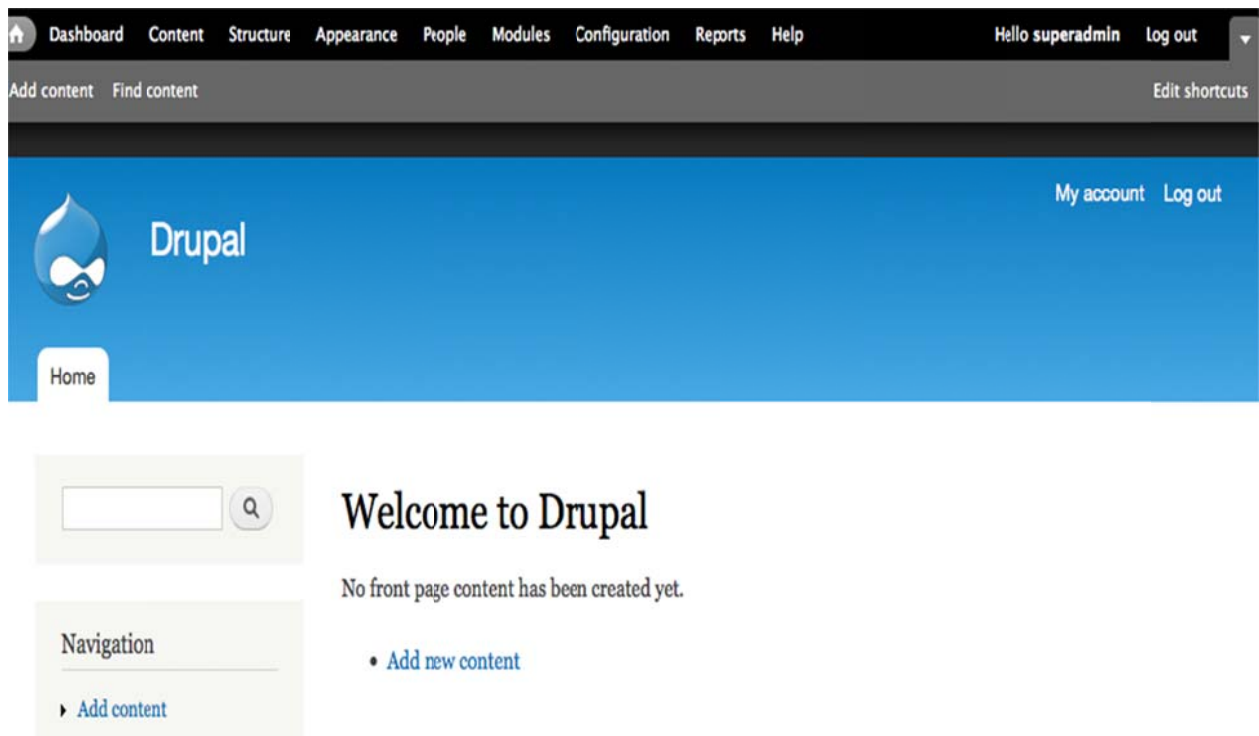


Figure A1.6: A new Drupal website, fresh from the installation process

## 8.2 Nodes

A node is a piece of content on a Drupal site. It can be an information page, a blog post or a press release. Content you don't usually read as separate web pages can also be nodes, such as images, videos, or containers to collect pages with restricted access for certain users.

As a Drupal developer, one of the most important things you will learn is how to use nodes to build the information structure on a website. This chapter explains how to use the basic features of nodes.

In Drupal 7, the term *node* has almost completely been replaced with *content*. But the term *node* is still widely used by many modules (plugins) and in documentation. This makes it important to know and recognize. In this manual, the terms *node* and *content* are used interchangeably if not otherwise stated.

---

### 8.2.1 Creating nodes

On the first page you see as an administrator on a new Drupal site, there are no less than three links to add new content – one in the sidebar, another in the shortcuts at the top of the page, and one right in the middle of the page. (See figure 1.1) They all open the administrative overlay, allowing you to create either an *Article* or a *Basic page*. (See figure 1.2)

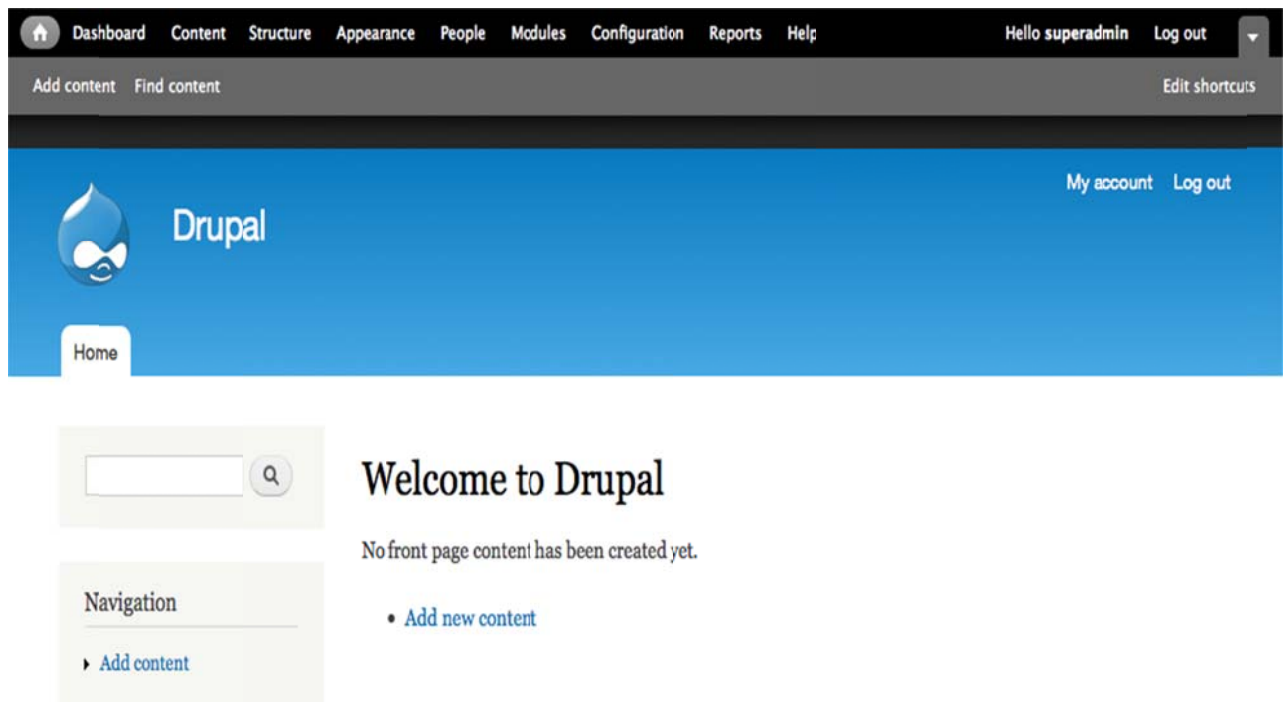


Figure 1.1: A Drupal standard installation without any content – click *add content* to create your first node!

Articles and Basic pages are two *node types* or types of content. Clicking on any of the type names will give you a form used to build a content of that type. This is called the node form. The forms used for Articles and Basic pages are different, but they work the same way. You get a number of fields where you can input information, and you have buttons for previewing or saving the piece of content. (See figure 1.3)

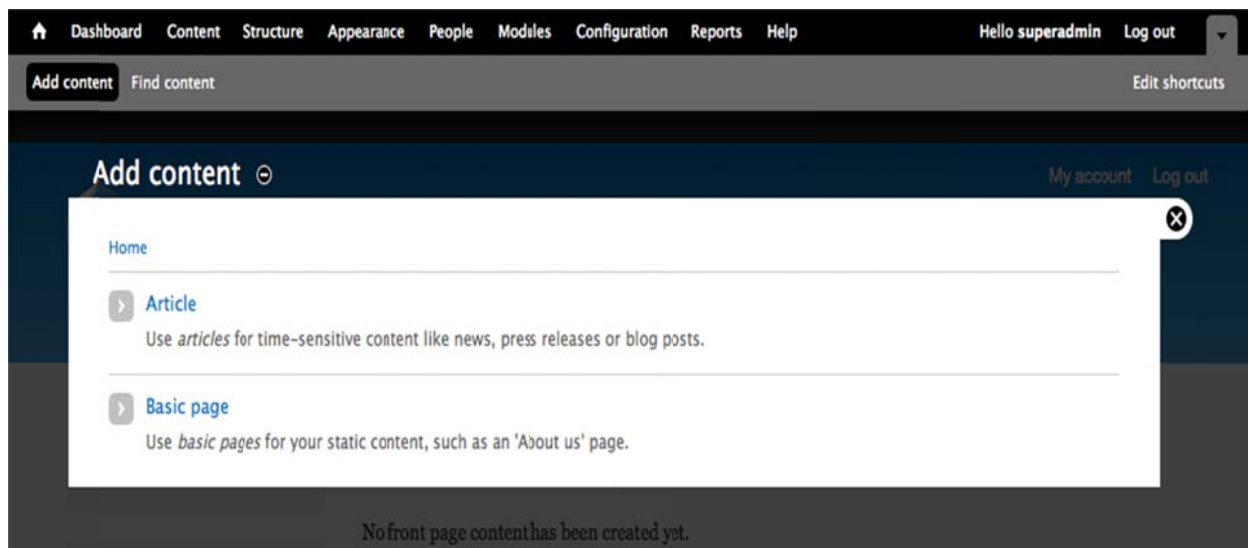


Figure 1.2: By default, nodes – content – come in two flavors: Articles and Basic pages.

Below is a description for the fields available in the Article node form.

- **Title:** This is the headline for the article, and will be shown at the top of the article's page on the website (it will also be used for the HTML title shown in the web browser's top bar).
- **Tags:** This is an opportunity to assign your article one or more keywords, which are used to categorize the content. Drupal will suggest any matching existing keywords while you write, but you can also provide new ones. If you want several keywords, separate them with commas. Keywords are usually displayed as links to lists of all nodes with the same keyword.
- **Edit summary/hide summary:** When clicking *edit summary*, you get a box where you can write a summary of the article. Summaries are often used when articles are listed, as a so-called *teaser*. The link *hide summary* hides the box again. If no explicit summary is written, Drupal will create one from the first part of the body.
- **Body:** This field is used for the main text of the article.
- **Text format:** Usually the body field only contains plain text, but it can interpret some HTML markup. The settings in the text format field decide which markup should be allowed, which is important from a security point of view.
- **Image:** This gives you the opportunity to upload an image to be displayed along with the article. Uploaded images can by default be provided with alt texts – text shown if the image isn't loaded (this is important for screen readers used by the visually impaired, as well as for search engine robots).
- At the bottom of the node form, there are a number of settings for menus, comments, and some other things.
- Finally, there are buttons to save or preview the article. Clicking the *save button* takes you to a new web page that shows the article you just created.

**Title \***

Hello world!

**Tags**

alpha, beta, gamma

Enter a comma-separated list of words to describe your content.

**Summary (Hide summary)**

This is a special summary, created by clicking the 'edit summary' link.

Leave blank to use trimmed value of full text as the summary.

**Body**

Proin in velit libero, vitae sodales lectus. Aenean posuere interdum magna ut imperdiet. In mollis velit et ante posuere ut ullamcorper augue ultricies. Fusce eu sem ut lectus scelerisque dignissim. Pellentesque egestas orci suscipit elit tincidunt a egestas magna malesuada.


Nullam ut leo orci. Duis sed placerat sapien? Etiam aliquet ultricies dui; vitae rutrum risus sollicitudin ac. Vestibulum consectetur dolor orci aliquam.

**Text format** Filtered HTML ▾ [More information about text formats ?](#)

- Web page addresses and e-mail addresses turn into links automatically.
- Allowed HTML tags: <a> <em> <strong> <cite> <blockquote> <code> <ul> <ol> <li> <dl> <dt> <dd>

- Lines and paragraphs break automatically.

**Image**

 [DSCN4861.JPG](#) (589.13 KB) [Remove](#)

**Alternate text**

This text will be used by screen readers, search engines, or when the image cannot be loaded.

Upload an image to go with this article.

**Menu settings**  
Not in menu ☐ Provide a menu link

**Revision information**  
No revision

**URL path settings**  
No alias

**Comment settings**  
Open

**Authoring information**  
By superadmin

**Publishing options**  
Published, Promoted to front page

[Save](#) [Preview](#)

Figure 1.3: The form used to create new articles has room for different types of information.

---

## 8.2.2 Editing nodes and managing revisions

An article page, like other node pages, has two tabs: *view* and *edit*. By clicking the edit tab, you will reopen the node edit form, which is identical to the one you just used, except for two things. The form is prepopulated with the content of your article, and there is a button to delete the node next to the *save* and *preview* buttons.

Among the settings at the bottom of the node form, you will find *revision information*. (See figure 1.5) Checking the option *create new revision* tells Drupal to archive the present version of the node when a new version is created – a pretty useful feature if you want version control of your content. A node that has archived revisions will be displayed with an additional tab – *revisions*. A click on the tab gives you an overview of all available revisions, with links to view them, to revert the node to a selected revision, and also to delete revisions, if necessary. (See figure 1.6) Reverting a node doesn't delete any revisions – it merely places a copy of the selected revision on top as the current revision.

Menu settings  
Not in menu

Revision information  
New revision

URL path settings  
No alias

Comment settings  
Open

Authoring information  
By superadmin on 2011-01-03 15:00:55 +0100

Publishing options  
Published, Promoted to front page

☒ Create new revision  
  

Revision log message

Changed the article body.

Provide an explanation of the changes you are making. This will help other authors understand your motivations.

Save

Preview

Delete

Figure 1.5: Drupal has built-in functionality for node revisions.

Dashboard Content Structure Appearance People Modules Configuration Reports Help

Hello superadmin Log out

Add content Find content Edit shortcuts

Revisions for Hello world!

VIEW EDIT REVISIONS

Home » Hello world!

Revisions allow you to track differences between multiple versions of your content, and revert back to older versions.

REVISION	OPERATIONS
01/03/2011 - 15:12 by superadmin Changed the article body.	current revision
01/03/2011 - 15:00 by superadmin	revert delete

Figure 1.6: The tab *revisions* shows all previous versions of a node, along with any log messages for the revisions.

As a rule, links and settings are only visible if you are allowed to use them. Thus, the tab for editing a node is only displayed for users who are allowed to edit it. The option for changing text formats is only displayed if you are allowed to switch formats. When logged in with user account 1 (the administrator account), you will usually see all settings, for better or for worse.



---

### 8.2.3 Other node settings

Apart from revision information, the bottom of the node form also usually contains five other tabs with settings.

- Menu settings: This gives you the chance to add a menu item linking to the node.
- URL path settings: This allows you to give the node a URL path in parallel to the path used internally by Drupal. The internal path for nodes is always in the form 'node/NN', where NN is a unique ID number for the node.
- Comment settings: This allows you to turn commenting on or off on each node. If the node already has comments, you can choose to hide them.
- Authoring information: This shows which user account was used to create the node, and when it was created. Both fields can be changed if necessary.
- Publishing options: This gives you three settings controlling how and where the node should be displayed on the site.
  - Published: By default, only administrators may view unpublished content, while published content is accessible for anyone.
  - Promoted to front page: With this option checked, the node will be included in the list that is used by default as Drupal's home page. (Most larger sites have customized home pages on which this option is hidden.)
  - Sticky at top of lists: This option makes the node show up above other nodes on the front page, as well as above some other lists.

---

### 8.2.4 View modes for nodes

When visiting the page for a node, you will usually see all of its content. When the node is listed on the front page, only parts of it are displayed. With articles, the body is replaced by a summary, and any images are displayed in a smaller format. This represents two view modes for nodes: *full node* and *teaser*.

Nodes may be configured to be displayed in different ways in each display mode.

---

### 8.2.5 Node types and node administration

Articles and basic pages are two different node types – two different templates used to create and manage the nodes. These differences are reflected in certain ways. For example:

- There are separate links for creating each node type.
- Each node type has room for different sets of information.
- You may set separate permissions for which users may create, edit and delete each node type.
- Each node type may have different default settings for comment handling, publishing options, menu links, and more. More details can be found below.

#### Default settings for node types

In the administration toolbar – the black list at the top of the page in a standard Drupal installation – there is an item called *Structure*. It leads to a page with some of the most interesting settings when building a Drupal site – such as *Content Types*. Clicking this link leads to an overview of all node types available on the website, along with links to manage each content type. Above the list, there is also a link to create new content types. (See figure 1.7)

[Home](#) » [Administration](#) » [Structure](#)

[+ Add content type](#)

NAME	OPERATIONS
<b>Article</b> (Machine name: article) Use <i>articles</i> for time-sensitive content like news, press releases or blog posts.	<a href="#">edit</a> <a href="#">manage fields</a> <a href="#">manage display</a> <a href="#">delete</a>
<b>Basic page</b> (Machine name: page) Use <i>basic pages</i> for your static content, such as an 'About us' page.	<a href="#">edit</a> <a href="#">manage fields</a> <a href="#">manage display</a> <a href="#">delete</a>

Figure 1.7: The administration link structure: content types in the administration toolbar provide an overview of all node types on your website.

The links named *Manage Fields* and *Manage Display* are used for managing and displaying fields in the node, *delete* link is used to delete the node type. The *edit* link is used to set the most basic properties for a node type (see figure 1.8). These properties are:

- **Name:** This is the name of the node type. Based on its plain-text name, Drupal suggests a machine name which is used to identify the node type in Drupal's database.
- **Description:** This text is shown in some lists of node types, such as the list provided when you click *add content*.
- **Submission form settings:** This gives you the opportunity to change the label used for the title of the node type – for example, if you have a node type for contacts, setting the label to 'name' makes more sense than 'title'. There are also options for changing the node preview settings, and for providing the node form with help text.
- **Publishing options:** This is used to change the default settings for the nodes' publishing states. This includes the option to create new revisions by default when editing content. Changing these settings will not affect any existing nodes (with the exception of revisioning, which is returned to default every time a node is edited).
- **Display settings:** This gives you the opportunity to show or hide information about who created the node and when.
- **Comment settings:** This allows a number of settings for comments, such as whether or not commenting is allowed by default, and whether comments are listed as a straight list or in a tree structure.
- **Menu settings:** These options dictate which menus should be able to link to nodes of this type, and whether nodes should be placed under a particular menu item by default.



**Name \*** Machine name: article [\[Edit\]](#)

The human-readable name of this content type. This text will be displayed as part of the list on the *Add new content* page. It is recommended that this name begin with a capital letter and contain only letters, numbers, and spaces. This name must be unique.

**Description**

Use `<em>articles</em>` for time-sensitive content like news, press releases or blog posts.

Describe this content type. The text will be displayed on the *Add new content* page.

**Submission form settings**

Title

**Publishing options**

Published , Promoted to front page

**Display settings**

Display author and date information.

**Comment settings**

Open, Threading , 50 comments per page

**Menu settings****Default options**☒ Published☒ Promoted to front page☐ Sticky at top of lists☐ Create new revision

Users with the *Administer content* permission will be able to override these options.

[Save content type](#)[Delete content type](#)

Figure 1.8: It is possible to set a number of default settings per content type, such as node revisioning.

## Node administration

Nodes may be scattered all over a Drupal site; it is not always easy to find the exact node you are looking for. The administration toolbar's *Content* option provides a list of all nodes on a website, along with some useful tools (see figure 1.9) including:

- Filters to limit the node list to only selected content types or nodes, with selected publishing options.
- Links to see, edit, and delete each node.
- Options for performing mass updates on nodes - for example, publishing, unpublishing, or deleting several nodes at once.

[+ Add content](#)

**SHOW ONLY ITEMS WHERE**

status

type

**UPDATE OPTIONS**

<input type="checkbox"/>	TITLE	TYPE	AUTHOR	STATUS	UPDATED	OPERATIONS
<input type="checkbox"/>	Hello world!	Article	superadmin	published	01/03/2011 - 15:12	<a href="#">edit</a> <a href="#">delete</a>

Figure 1.9: The administration page to manage content has tools for performing mass updates on nodes.

Larger Drupal sites usually have custom tailored administration pages that manage content according to workflows relevant to that site. The Views Bulk Operations module offers an alternative administration page for content, with more options and greater flexibility.

---

### 8.2.6 Node comments

When the *Comment* module is enabled, which is the case for a standard installation, users may post comments to nodes. (See figure 1.10) As previously mentioned, it is possible to turn the comment settings on and off for each node, and it is also possible to change the default setting for each node type.

Administrators can manage comments in two different ways:

- Each comment has an edit link, allowing administrators to edit the content of each comment, including changing posting information and optionally publishing/unpublishing the comment. (See figure 1.11)
- On the content overview page there is a Comments tab that leads to a list of all published comments on the website. There are also subtabs available to switch between viewing published and unpublished comments. Each list has tools for publishing, unpublishing and deleting comments. (See figure 1.12 and 1.13)

Comments are similar to nodes in their structure, but from a technical point of view, they are not nodes.

The screenshot displays a Drupal 7 administrative interface. At the top is a navigation bar with links: Dashboard, Content, Structure, Appearance, People, Modules, Configuration, Reports, and Help. On the right, it says 'Hello superadmin' and 'Log out'. Below the navigation bar is a sub-header with 'Add content' and 'Find content' links, and an 'Edit' link on the right. A sidebar on the left contains a link to 'Add content'. The main content area shows a list of tags: 'alpha', 'beta', and 'gamma'. Below the tags, there is a section titled 'Comments'. It contains three comment entries, each with a user profile (superadmin), a timestamp (Wed, 01/26/2011 - 09:47, 09:48, 09:49), a permalink, and the comment text. The first comment is 'A first comment' with the text 'This is a comment.' and actions 'delete', 'edit', and 'reply'. The second comment is 'A reply to the first comment' with the text 'This is a comment to the first comment' and actions 'delete', 'edit', 'reply', and 'approve'. The third comment is 'A second comment' with the text 'This is a comment, too' and actions 'delete', 'edit', and 'reply'. Below the comments is a section titled 'Add new comment'. It has a 'Your name' field with the value 'superadmin', a 'Subject' field, and a 'Comment' field. Below the comment field is a 'Text format' dropdown menu set to 'Filtered HTML', with a link to 'More information about text formats'. Below the text format is a list of allowed HTML tags: <a>, <em>, <strong>, <code>, <ul>, <ol>, <li>, <dl>, <dt>, <dd>. At the bottom are 'Save' and 'Preview' buttons.

Figure 1.10: Nodes may have comments, and comments may be ordered in a tree structure.

Navigation

- Add content

Administration

Authorized by: superadmin

E-mail:

The content of this field is kept private and will not be shown publicly.

Homepage:

Authorized on: 2011-01-26 09:47 +0100

Status: ☒ Published ☐ Not published

Subject: A first comment

Comment \*  
This is a comment.

Figure 1.11: Administrators can, if necessary, edit each individual comment.

Content

CONTENT COMMENTS

Home » Administration » Content

Published comments Unapproved comments (0)

UPDATE OPTIONS

Unpublish the selected comments

<input type="checkbox"/>	SUBJECT	AUTHOR	POSTED IN	UPDATED	OPERATIONS
<input type="checkbox"/>	A reply to the first comment	superadmin	An example article	01/26/2011 - 09:49	<a href="#">edit</a>
<input type="checkbox"/>	A second comment	superadmin	An example article	01/26/2011 - 09:49	<a href="#">edit</a>
<input type="checkbox"/>	A first comment	superadmin	An example article	01/26/2011 - 09:47	<a href="#">edit</a>

Figure 1.12: The administration list for published comments has tools to unpublish several comments at once.

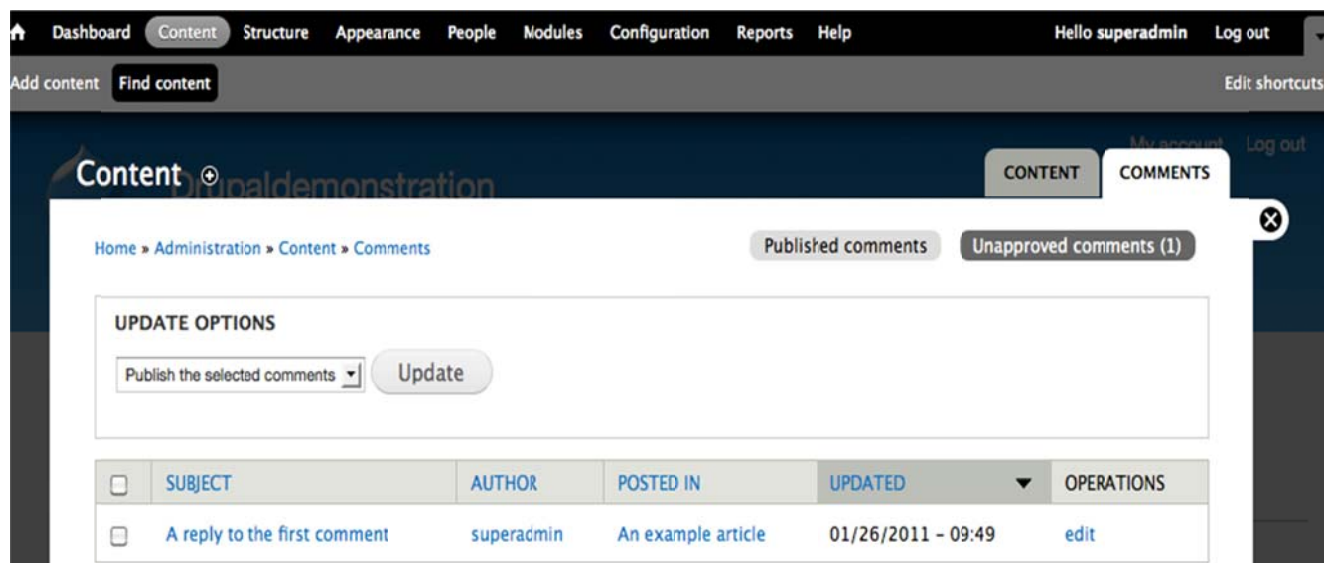


Figure 1.13: The list of unpublished comments has tools similar to the list of published comments.

## 8.3 Users and permissions

Next to content, users are probably the most important component of a website; they might even be more important than content. This section will show you how to manage users, divide them into groups, and decide the things that each group of users will be allowed to do on your website.

### 8.3.1 Adding and managing users

The administration toolbar's people link provides you with a list of all user accounts registered on your website. (See figure 2.1) The list of users shares many features with the content list. There are links to view and edit each account, and there are tools for filtering and mass updating accounts.

[+ Add user](#)

**SHOW ONLY USERS WHERE**

role	<input type="text" value="any"/>
permission	<input type="text" value="any"/>
status	<input type="text" value="any"/>

**UPDATE OPTIONS**

<input type="checkbox"/>	USERNAME	STATUS	ROLES	MEMBER FOR	LAST ACCESS	OPERATIONS
<input type="checkbox"/>	superadmin	active	• administrator	4 hours 15 min	3 sec ago	<a href="#">edit</a>

Figure 02.1: The list of user accounts contains shortcuts to edit each account, as well as tools for mass updates.

On top of the list is a link *add user*, used to add new accounts to the site. The form for adding new accounts is very similar to the form you get when editing an existing account. (See figure 2.2) Below is a description of the settings in the forms.

- Username: This is the user's name on the website, used when logging in. It must be unique.
- Current password (only when editing your own account): To change your e-mail address or password, you must usually give your current password.
- E-mail address: Like the username, the e-mail address must be unique. The reason for this is that the e-mail address should be available to use if the password or username are forgotten.
- Password/confirm password: This is the user's password, repeated to avoid misspellings. Note that Drupal *cannot* show the current password. All passwords are encrypted before they are stored and Drupal has no way of decrypting them. If you enter a short or simple password, you will get a warning, but Drupal won't prevent you from using it.
- Status: User accounts that are *blocked* cannot be used for logging in.
- Roles: This shows or sets which *permission roles* the user has. (See next section for details.) All users automatically have the role of *authenticated user* as soon as they log in.
- Picture (only on editing): This allows users to upload an image and associate it with their account.
- Administrative overlay (only on editing): This setting makes it possible to disable the administrative overlay for selected users, displaying the administration pages without the overlay.
- Locale settings (only on editing): This setting is used to change the time zone for a user.
- Notify user of new account (only on creation): This option makes Drupal send out an e-mail to the new user with the account information. This is the only time Drupal



sends out a password – if users lose their passwords, they will get a one-time login, rather than their existing password.

**Username \***

superadmin

Spaces are allowed; punctuation is not allowed except for periods, hyphens, apostrophes, and underscores.

**Current password**

Enter your current password to change the *E-mail address* or *Password*. [Request new password](#).

**E-mail address \***

johan.falk@nodeone.se

A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.

**Password**

**Password strength:**

**Confirm password**

To change the current user password, enter the new password in both fields.

**Status**

☐ Blocked

☒ Active

**Roles**

☒ authenticated user

☒ administrator

**PICTURE**

**Upload picture**

Bladdra...

Your virtual face or picture. Pictures larger than 1024x1024 pixels will be scaled down.

▼ **ADMINISTRATIVE OVERLAY**

☒ Use the overlay for administrative pages.

Show administrative pages on top of the page you started from.

▼ **LOCALE SETTINGS**

Your time zone setting will be automatically detected if possible. Confirm the selection and click save.

**Time zone**

Europe/Stockholm: Monday, January 3, 2011 - 17:04 +0100

Select the desired local time and time zone. Dates and times throughout this site will be displayed using this time zone.

Save

Figure 2.2: The page used to edit user accounts contains a number of settings, such as which permissions/roles the user has.

Best practice dictates that you should not use the first user account as a personal account. The first user account bypasses all access controls in Drupal, and should be available to pass on to new site managers.

---

### 8.3.2 Permissions and roles

The first account created on a Drupal site, user 1, has permission to do *everything* on the site, but what about other users?

To see which permissions different types of users have – and to change these settings – you use the tab *Permissions* under the *People* link in the toolbar. The resulting page contains a long list of permissions, and there is one column for each *permission role* on your site – anonymous users, authenticated users and administrators. (See figure 2.3)

By checking or unchecking the different permissions, it is possible to do things like give authenticated users permission to create articles or allow anonymous users to search the site. The permissions are grouped by the module that is responsible for them, and the more modules that are installed, the more permissions are sure to turn up.



[Home](#) » [Administration](#) » [People](#)

Permissions

Roles

Permissions let you control what users can do and see on your site. You can define a specific set of permissions for each role. (See the [Roles](#) page to create a role). Two important roles to consider are Authenticated Users and Administrators. Any permissions granted to the Authenticated Users role will be given to any user who can log into your site. You can make any role the Administrator role for the site, meaning this will be granted all new permissions automatically. You can do this on the [User Settings](#) page. You should be careful to ensure that only trusted users are given this access and level of control of your site.

[Hide descriptions](#)

PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR
<b>Block</b>			
Administer blocks	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<b>Comment</b>			
Administer comments and comment settings	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
View comments	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Post comments	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Skip comment approval	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<b>User</b>			
Administer permissions <i>Warning: Give to trusted roles only; this permission has security implications.</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Administer users <i>Warning: Give to trusted roles only; this permission has security implications.</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
View user profiles	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Change own username	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Cancel own user account <i>Note: content may be kept, unpublished, deleted or transferred to the Anonymous user depending on the configured <a href="#">user settings</a>.</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Select method for cancelling own account <i>Warning: Give to trusted roles only; this permission has security implications.</i>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 2.3: The permissions list contains a large matrix of settings.

The *Roles* tab, visible on the permissions list, shows you a list of all permission roles available on the website. (See figure 2.4) If your site needs more permission levels than the three included in a standard installation, which is very likely, this is the place to add more. Each new role will be represented by a new column in the permissions matrix, and you may set permissions for each role independent of the others. (See figure 2.5) Quick and slick!

Assigning roles to users is done in one of two ways – you can either edit each user account you want to change, or you can make mass updates from the user list.

[Home](#) » [Administration](#) » [People](#) » [Permissions](#)

Permissions

Roles

Roles allow you to fine tune the security and administration of Drupal. A role defines a group of users that have certain privileges as defined on the [permissions page](#). Examples of roles include: anonymous user, authenticated user, moderator, administrator and so on. In this area you will define the names and order of the roles on your site. It is recommended to order your roles from least permissive (anonymous user) to most permissive (administrator). To delete a role choose "edit role".

By default, Drupal comes with two user roles:

- Anonymous user: this role is used for users that don't have a user account or that are not authenticated.
- Authenticated user: this role is automatically granted to all logged in users.

[Show row weights](#)

NAME	OPERATIONS
 anonymous user <i>(locked)</i>	<a href="#">edit permissions</a>
 authenticated user <i>(locked)</i>	<a href="#">edit permissions</a>
 administrator	<a href="#">edit role</a> <a href="#">edit permissions</a>
<input type="text" value="editor"/>	<input type="button" value="Add role"/>

Figure 2.4: You may add more permission roles in the roles list.

Home > Administration > People

Permissions let you control what users can do and see on your site. You can define a specific set of permissions for each role. (See the [Roles](#) page to create a role). Two important roles to consider are Authenticated Users and Administrators. Any permissions granted to the Authenticated Users role will be given to any user who can log into your site. You can make any role the Administrator role for the site, meaning this will be granted all new permissions automatically. You can do this on the [User Settings](#) page. You should be careful to ensure that only trusted users are given this access and level of control of your site.

[Hide descriptions](#)

PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR	EDITOR
<b>Block</b>				
Administer blocks	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<b>Comment</b>				
Administer comments and comment settings	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
View comments	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Post comments	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Skip comment approval	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 2.5: Each new role can have separate permission settings.

If a user has more than one role, which is quite possible, she will get permissions based on *all* these roles. The rule is that roles decide what you *can* do, not what you can't.

The permissions matrix is one of the busiest settings pages in Drupal, but fortunately, each permission is more or less self-explanatory. Since new modules often add new permissions, it is common to wait until a project is nearly finished before setting all permissions. This avoids having to go through the whole list more times than is necessary.

### 8.3.3 Other user account settings

Selecting the *Configuration* link from the toolbar and then *Account Settings* in the *People* block on the Configuration Page will show you a few more user settings that a Drupal developer should know. The most important are described below.

- Administrator role: The role set here will automatically have all permissions set in the permissions matrix.
- Registration and cancellation: This setting determines how new accounts should be created - for example, if visitors should be allowed to sign up, and also how content and accounts should be treated when an account is cancelled.
- Personalization – signatures: If signatures are enabled, users will be allowed to set a signature that will be added to all their comments (but not to other content). If a user changes her signature, signatures on existing comments are also affected.

- E-mails: This setting contains a number of e-mail templates that are used when users register, if they lose their passwords, and in some other cases. Note that there are some *token replacement patterns* available for dynamic replacements, such as [user:name] and [user:one-time-login-url].

## 8.4 Regions and blocks

When you visit a node page in Drupal – or any other page – Drupal pulls out the content corresponding to the current URL and formats it to make it possible for a web browser to display. On the URL *node/1*, for example, Drupal displays the content for the node with ID 1. But it is not just the content of node one that is displayed; there may also be elements like menus, search forms, related content, latest comments on the site, and much more.

Both the main content and the other elements are displayed as *blocks*, placed in one of the *regions* of the website. These blocks can be moved around, and there are basic, as well as more sophisticated, ways of adding new blocks to your website.

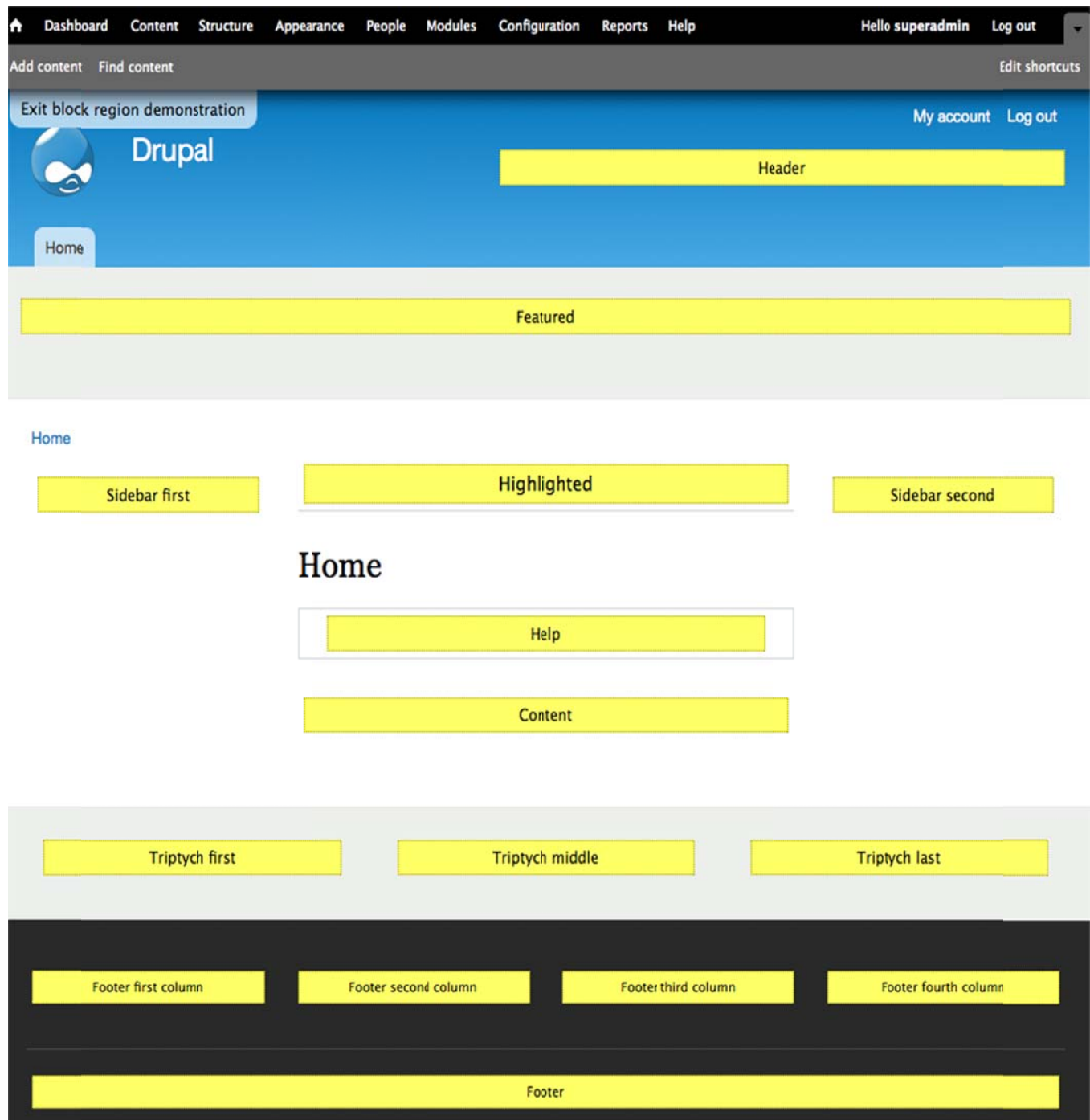


Figure 3.1: Blocks are placed in one of the regions of the website. The *Demonstrate block regions* link on the administration page for blocks gives an overview of the regions available in your current theme.

You can reach a list of all available blocks on the site by using the administration toolbar to select *Structure* and then *Blocks*. (See figure 3.2) The blocks are grouped by the region where they are placed – which could, for example, be *Sidebar first* or *Content*. There is also a list of blocks under the subheading *Disabled*, meaning they don't show up at all. You can move blocks to a new region by using the select list of region names, or by simply clicking on a block's sorting arrow and dragging it to another region's subheader.

Above the list of blocks is the link *Demonstrate block regions*. It leads to a page where all regions are printed clearly on an empty site template, making it easy to get an overview of the available regions. (See figure 3.1) Don't miss the *Exit block region demonstration* link to return to the block list.



This page provides a drag-and-drop interface for assigning a block to a region, and for controlling the order of blocks within regions. Since not all themes implement the same regions, or display regions in the same way, blocks are positioned on a per-theme basis. Remember that your changes will not be saved until you click the *Save blocks* button at the bottom of the page. Click the *configure* link next to each block to configure its specific title and visibility settings.

[Demonstrate block regions \(Bartik\)](#)

[+ Add block](#)

[Show row weights](#)

BLOCK	REGION	OPERATIONS
<b>Header</b>		
<i>No blocks in this region</i>		
<b>Help</b>		
<a href="#">+ System help</a>	Help	<a href="#">configure</a>
<b>Highlighted</b>		
<i>No blocks in this region</i>		
<b>Featured</b>		
<i>No blocks in this region</i>		
<b>Content</b>		
<a href="#">+ Main page content</a>	Content	<a href="#">configure</a>
<b>Sidebar first</b>		
<a href="#">+ Search form</a>	Sidebar first	<a href="#">configure</a>
<a href="#">+ Navigation</a>	Sidebar first	<a href="#">configure</a>
<a href="#">+ User login</a>	Sidebar first	<a href="#">configure</a>
<b>Disabled</b>		
<a href="#">+ Main menu</a>	- None -	<a href="#">configure</a>
<a href="#">+ Management</a>	- None -	<a href="#">configure</a>
<a href="#">+ Recent comments</a>	- None -	<a href="#">configure</a>
<a href="#">+ Recent content</a>	- None -	<a href="#">configure</a>
<a href="#">+ Shortcuts</a>	- None -	<a href="#">configure</a>
<a href="#">+ Syndicate</a>	- None -	<a href="#">configure</a>
<a href="#">+ User menu</a>	- None -	<a href="#">configure</a>
<a href="#">+ Who's new</a>	- None -	<a href="#">configure</a>
<a href="#">+ Who's online</a>	- None -	<a href="#">configure</a>

[Save blocks](#)

Figure 3.2: The administration page for blocks shows all the blocks available on the site, grouped by the regions in which they are placed.

### 8.4.1 Block settings

In the block list, there is a *configure* link available for each block. Each link leads to a page where you may change the settings for the block (see figure 3.3).

- Block title: This can be used for overriding the title of the block as it is shown to users. To hide the title, enter *<none>*.
- Region settings: This is an alternative way of moving the block between regions, similar to the select lists in the block overview list.
- Visibility settings: These settings provide some basic options for determining when the block should be visible or not:
  - Pages: This allows for showing or hiding the block based on the URL of the viewed page. You may use \* as a wildcard to replace the entire URL or just parts of it. Any pattern will be compared with both the internal paths ('node/1') and URL aliases ('information/about-us').
  - Content types: This can be used to show the block only when selected node types are viewed.
  - Roles: This can be used to show the block only to users with the selected roles.
  - Users: Enabling this setting allows users to determine if the block should be visible or not. (These blocks then become available on each user's account edit page.)

Apart from these settings, every block also has its own particular settings. The block for most recent comments, for example, has settings for how many comments should be displayed.



Block title

Override the default title for the block. Use `<none>` to display no title, or leave blank to use the default block title.

REGION SETTINGS

Specify in which themes and regions this block is displayed.

Bartik (default theme)

Seven (administration theme)

Visibility settings

<p><b>Pages</b> Not restricted</p> <p><b>Content types</b> Not restricted</p> <p><b>Roles</b> Not restricted</p> <p><b>Users</b> Not customizable</p>	<p><b>Show block on specific pages</b></p> <p><input checked="" type="radio"/> All pages except those listed</p> <p><input type="radio"/> Only the listed pages</p> <div style="border: 1px solid #ccc; height: 100px; margin-top: 10px;"></div> <p>Specify pages by using their paths. Enter one path per line. The '*' character is a wildcard. Example paths are <i>blog</i> for the blog page and <i>blog/*</i> for every personal blog. <i>&lt;front&gt;</i> is the front page.</p>
---	--

Save block

Figure 3.3: Every block has its own settings - for example, some settings determine in which context the block should be shown or hidden.

## 8.4.2 Adding blocks

Many modules provide the block list with new blocks, and there are also a lot of modules which let you, as an administrator, create new blocks through configuration. To create the most basic blocks, though, you only need the Block module itself and the *add block* link right above the block list. This leads to a page where you can create a block with static content.

---

### 8.4.3 Complements and alternatives to blocks

Block management can quickly become a mess on large and complex Drupal sites. In response to this problem, a number of modules have been created. Here are the two most important to know:

- Context: Among other things, this module helps set visibility rules for blocks in a more flexible way.
- Panels and Page manager (part of Chaos tools suite): These modules replace the regular block system with *panel panes*. Compared to blocks, these panes make it far easier to access and use contextual information.

These modules provide more functionality by complementing or replacing the block system. Discussion on Context module, Panels and Page manager are beyond the coverage of this manual.

## 8.5 Menus

By default, content on a Drupal site is not automatically placed in any particular structure. When creating a node, you don't choose *where* on the site it should be. You create it, and then other parts of Drupal can make it appear as a subpage to a particular menu item, in a list in a particular section, or as a part of another structure.

The most direct way of bringing structure to your Drupal site is to use menus. These are links collected in a tree structure.

---

### 8.5.1 Displaying menus

A standard installation of Drupal has four initial menus: *main menu*, *management*, *navigation* and *user menu*. More menus can be added via Drupal's interface, and you can also choose where and how they should be displayed.

There are, in principle, two ways of displaying menus:

- Each menu on the site has its own block, which can be placed in a region just like any block.
- The theme on the site can (but does not always) have two places where menus are displayed in a special format – *main links* and *secondary links*. In a standard Drupal installation, the main links are displayed as large white tabs against the blue header, while the secondary links are displayed as discrete links in the upper-right corner of the site.

Which menus should be used for *main links* and *secondary links* can be changed at the toolbar's *Structure* link, in the *Menu* and *Settings* tabs.

The display of main links and secondary links only hold one level of menu links. Submenu items are not shown. It is possible to use secondary links to display subitems of the primary menu by configuring them to fetch links from the same menu. The Menu block module provides further possibilities to display selected levels and parts of a menu.

## 8.5.2 Creating and editing menu links

As with many other administration tasks, there is an overview for managing menus. It can be found by going to the toolbar and selecting first *Structure* and then *Menus*, and it displays all the menus available on your site. (See figure 4.1) Each menu presents three options.

- List links: This gives you a list of all items in this menu, and is usually what you want to do when managing a menu.
- Edit menu: This allows you to change the name and description of the menu itself (not the links it contains). You may also delete any menus you have created yourself.
- Add link: This leads to a page for adding another link in the menu. See details below.

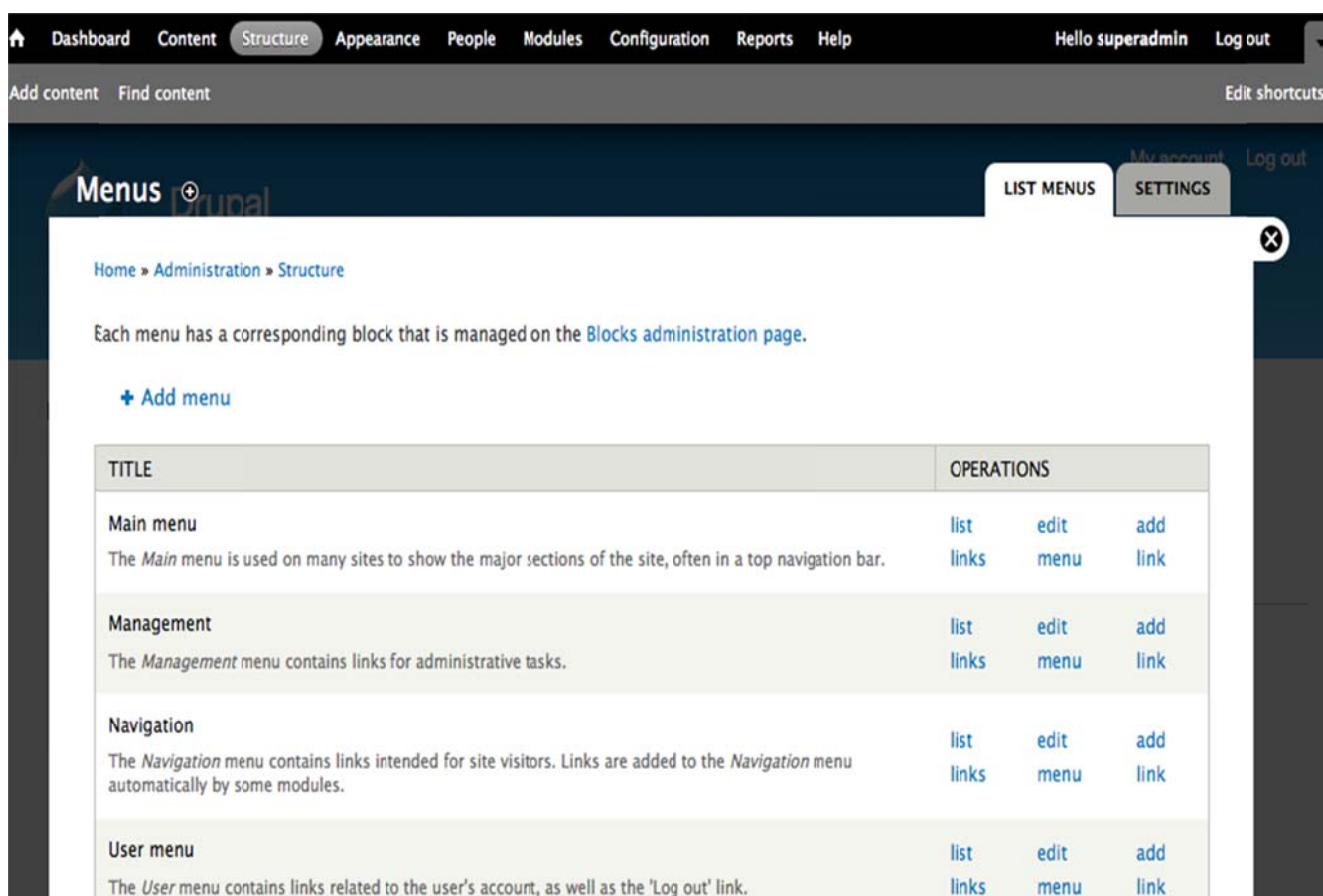


Figure 4.1: The menu overview can be found in Structure, Menus.

At the top of the list of menus is an *Add menu* link used for adding further menus. The only difference between menus that you create yourself and those provided by modules (or the standard installation) is that your custom menus can be deleted.

The page listing menu links allows you to manage the content of the menu in a few different ways (see figure 4.2):

- You may change the menu structure by clicking and dragging the sorting arrows. Submenu items are created by indenting a menu item.
- You may enable or disable a menu item with the checkbox *enabled*. Disabled items won't be displayed in the menu, and any child items will also be hidden. But the pages they lead to are not affected.

- You may edit each menu item using the *edit* link. This leads to a page similar to the one used for creating new items.
- Each item managed by Drupal's Menu module also has a *delete* link. Links defined by other modules are managed by their respective module settings, but can still be disabled in the menu link list.

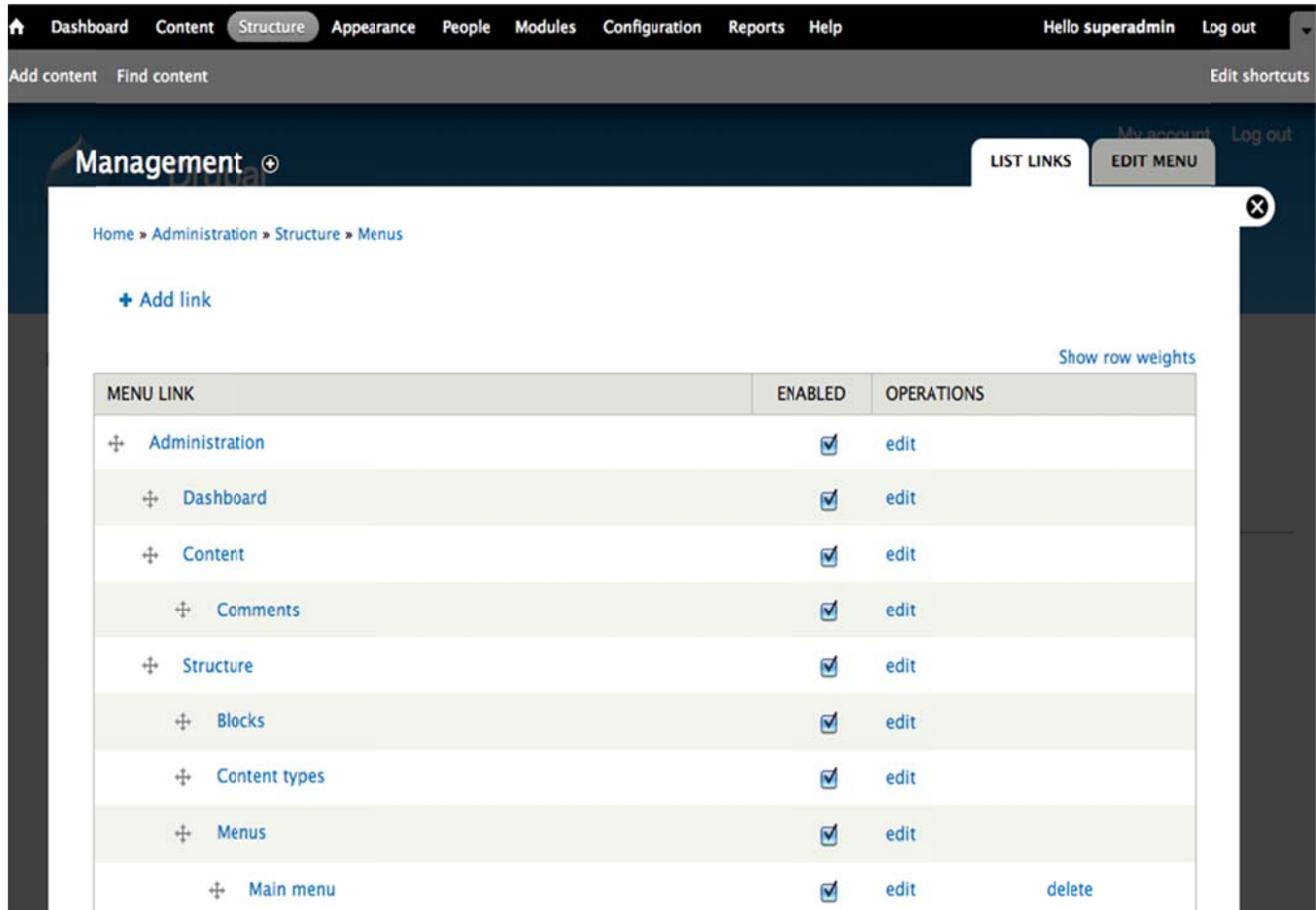


Figure 4.2: Each menu has a list of all links included in that menu.

Right above the list of menu items there is an *Add item* link, which is used to add new links to the currently viewed menu. The form for creating or editing menu items has the following information:

- Menu link title: This is the clickable text that will be displayed to users.
- Path: This is the URL the menu link leads to. When creating menu links, it is strongly recommended to use the internal path ('node/1') rather than absolute paths ('http://example.com/node/1'), since using internal paths makes it possible to move the site without breaking links. It also makes it possible for Drupal to replace links with any URL aliases.
- Description: This is a tooltip that is usually shown when hovering over the menu item.
- Enabled: This corresponds to the enabled setting in the menu item list.
- Show as expanded: If this option is checked, all child items of this menu link will be loaded and displayed, even if the user has navigated to another part of the menu.
- Parent link: This setting dictates which menu link this item should be placed under, if any. This option can be replaced by manual click-and-drag sorting in the menu item list.

- **Weight:** This setting determines the sorting order for menu items with the same parent, with lower weight numbers floating to the top. This option can be overridden by manual click-and-drag sorting in the menu item list.

Menu links don't have to lead to pages on your Drupal site. They can point to any valid URL.

---

### 8.5.3 Creating menu links for nodes

A quick and easy alternative when creating menu links is to use the menu options available on node edit pages, under *Menu settings*. If the *Provide a menu link* option is checked, a number of new options become available (see figure 4.3). All settings are similar to the menu item configurations described in the previous section.

Figure 4.3: You can create menu links to nodes right from the node's edit form.

By default, articles and basic pages may only be placed in the *Main menu*. There are settings on each node type determining which menus should be available in the node edit form, as well as default settings for the *parent item*.

## 8.6 Other basic Drupal core settings

This section collects some miscellaneous administrative settings a Drupal developer should know.

---

### 8.6.1 Administration aids

There are a number of shortcuts and nifty tools enabled in a standard Drupal installation, all with the purpose of making site administration easier. They are summarized below.

- The black toolbar at the top on Drupal pages is provided by the *Toolbar* module. It contains links to the top-level items in the menu management.
- The administrative overlay, the frame that appears when clicking on administration links, is provided by the *Overlay* module. One of its points is to make it easier to find the page you were on when starting your administration.
- The *Contextual links* module adds a number of links in drop-down menus at various elements on your Drupal site. The menu is accessed by clicking on the gear icon that appears when hovering over elements that have contextual links – such as blocks.
- The *Dashboard* module makes it easier to build a collection of blocks on one single administration page (found under the dashboard link in the toolbar).
- The gray list just below the black toolbar is provided by the *Shortcut* module, which also provides links (marked with plus signs) to add default shortcuts. The latter are used to add links to the former, which is a handy way to always have your most-visited administration pages one click away. It is possible to create different sets of shortcuts. From the toolbar, select Configuration, Shortcuts. Users with relevant permissions may select which shortcut sets to use on a separate tab on their user pages. Note that while shortcuts remind us of menus, they are technically separated.

The modules Administration menu and Admin are popular complements to Drupal's built-in tools for administrators. They both provide alternative, and in some aspects more user-friendly, administration menus.

---

### 8.6.2 Text formats

In Drupal, all text that may contain markup is handled by one of the website's *text formats*. These are rules determining how the text should be processed before being displayed. The text formats have three main purposes:

- Security: The text formats make sure that any malicious code or script, entered by malicious users, doesn't have malicious effects on the site or its visitors.
- Sanitation: The inputted text is processed to make sure that any markup is clean and follows common standards. This is one of the reasons for Drupal being naturally search-engine friendly (as well as friendly towards screen readers).
- Comfort: Text formats may also be used to convert certain expressions to HTML. In Drupal core, this is used to automatically create things like line breaks and links from URLs, but it could also be used for allowing the same type of markup as is used on Wikipedia, for example.

Each text format consists of one or more *filters*. The formats and most filters can be managed using the toolbar *Configuration, Text formats*.

Even if Drupal processes all formatted text, the original text is never changed, which is an important rule for how to treat user input. Since the original potentially malicious input is stored in the database, it is important to keep in mind that user-provided text should always be sanitized.

---

### 8.6.3 Other settings

This last section of the Drupal core basics chapter contains some loosely related items you may find useful.



- The page you use as your site's home page can be set from the toolbar *Configuration, Site information*. The default setting is "node," which provides a list of teasers of nodes marked with *promote to front page*.
- Drupal depends on a number of scheduled activities being carried out on a regular basis – for example, allowing indexing of new content for searches. If your server has native cron functions, you can turn off Drupal's backup functionality for scheduled activities at the toolbar *Configuration, Cron*. On the same page, you can also trigger cron activities manually - for example, to make new content immediately searchable.
- If things start behaving weirdly while you're building your Drupal site, it is wise to empty Drupal's cache. This processed data is temporarily stored to speed up the website. If this doesn't help, check the error logs for interesting messages. You can clear the cache using the toolbar *Configuration, Performance*. The most interesting log messages are available at *Reports, Recent log messages*.
- When performing system updates on your site, you should put it in maintenance mode to make sure that site visitors aren't accessing the database while it is being updated. This setting is on the toolbar at *Configuration, Maintenance mode*.

## References

1. Anderson, R “*The Long and Winding Road to Web-Based Apps*”. Network Computing 19 March 2001: 79-84
2. Deitel, H.; et al., *Internet & World Wide Web How to Program*. Upper Saddle River, NJ: Prentice Hall, 2004
3. Chun, Russel. *Macromedia Flash Advanced*. US.: Peachpit Press, 2001
4. Bleyle, J., et al., *Macromedia Flash MX 2004 Actionscript Reference Guide*. [http://download.macromedia.com/pub/documentation/en/flash/mx2004/fl\\_actionscript\\_reference.pdf](http://download.macromedia.com/pub/documentation/en/flash/mx2004/fl_actionscript_reference.pdf), San Francisco, CA: Macromedia, Inc.,2003.
5. Curtis, H. *Flash Web Design: The art of Motion Graphics*. U.S.: New Riders Publishing, 2000.
6. Boumphrey, F., et al., *Beginning XHTML*. U.S.: Wrox Press Ltd, 2000.
7. Floyd, M. “Generating Style Sheets Dynamically.” *Webtechniques* January 2001:C2
8. “DHTML and CSS (For the World Wide Web),” Second Edition, Cranford Teague, Peachpit Press, 2001, ISBN 0-201-73084-7
9. “The Web Wizard’s Guide to Web Design,” James G. Lengel, Addison Wesley, 2001, ISBN: 0-201-74562-3
10. Drupal site, <https://www.drupal.org/>