



evropský
sociální
fond v ČR



EVROPSKÁ UNIE



MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



OP Vzdělávání
pro konkurenceschopnost



Slezská univerzita v Opavě

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Silesian University in Opava
School of Business Administration in Karviné

PORTAL AND ITS MANAGEMENT

Klepněte sem a zadejte text.

For the full-time study form

Petr Suchánek
Jan Górecki

Karviná 2014

OP VK No. CZ.1.07/2.2.00/28.0017 Project „Innovation of study programmes at Silesian university in Opava, School of Business Administration in Karvina“

Subject: Computer science.

Annotation: Currently, web portals can be considered as a key interface in the Internet. Web portals can be used as the mere presentation of content, as well as for business purposes such as online shops, access to corporate information systems, management of branch offices, communication and management support content within the intranet, etc. This study textbook is focused on the presentation of web portals development using technologies such as HTML, PHP, MySQL and Apache local server. The primary objective is to familiarize students with the creation of dynamic web sites, methods of work with scripts and linking dynamic websites with databases. After studying study textbook, students will be able to create simple portal solutions that will function both, with using the local server, and within the web hosting.

Key words: Web portal, script, database, web forms, data transmission.

© **Doplňí oddělení vědy a výzkumu.**

Author: **Doc. Mgr. Petr Suchánek, Ph.D.**
Ing. Jan Górecki

Recenzenti: Doc. RNDr. Petr Šaloun, Ph.D.
RNDr. Ing. Roman Šperka, Ph.D.

ISBN Doplňí oddělení vědy a výzkumu.

CONTENT

INTRODUCTION	7
1 WEB & INTERNET PORTAL	8
1.1 DEFINITION OF WEB PORTAL AND RELATED TERMS.....	8
1.2 TYPES OF WEB PORTALS.....	9
1.2.1 HORIZONTAL WEB PORTALS.....	9
1.2.2 VERTICAL WEB PORTALS.....	9
1.2.3 COMBINATION OF A HORIZONTAL AND VERTICAL WEB PORTAL.....	10
1.3 WEB PORTAL DEVELOPMENT	10
1.4 TOOLS AND TECHNOLOGIES FOR A WEB PORTAL DEVELOPMENT	12
1.5 PSPAD	13
2 HTML	15
2.1 CHARACTERISTICS OF HTML	15
2.2 CHARACTERISTICS OF XHTML	15
2.3 FUNDAMENTAL DIFFERENCES BETWEEN HTML AND XHTML	15
2.4 HTML TAGS AND ELEMENTS	15
2.5 HTML DOCUMENT	16
2.6 HTML ELEMENT SYNTAX	17
2.7 HTML PARAGRAPHS AND LINE BREAKS.....	17
2.8 HTML HEADINGS	18
2.9 HTML FONT	18
2.10 HTML LISTS.....	20
2.11 HTML IMAGES.....	22
2.12 HTML TABLES	23
2.13 HTML LINKS.....	27
2.14 FRAMES	29
2.14.1 CREATING FRAMES.....	29
2.14.2 THE <FRAMESET> ELEMENT ATTRIBUTES.....	30
2.14.3 LOADING CONTENT.....	31
2.14.4 THE <FRAME> ELEMENT ATTRIBUTES.....	31
2.14.5 FRAME'S NAME AND TARGET ATTRIBUTES	31
2.15 FORMS	32
2.15.1 TEXT INPUT CONTROLS.....	33
2.15.2 SINGLE-LINE TEXT INPUT CONTROLS	33
2.15.3 PASSWORD INPUT CONTROLS.....	34
2.15.4 MULTIPLE-LINE TEXT INPUT CONTROLS	34
2.15.5 CREATING BUTTON	35
2.15.6 CHECKBOXES CONTROL.....	35
2.15.7 RADIO BOX CONTROL.....	36
2.15.8 SELECT BOX CONTROL.....	36
2.15.9 FILE SELECT BOXES.....	37
2.15.10 HIDDEN CONTROLS.....	37
2.15.11 SUBMIT AND RESET BUTTON	37
3 CSS (CASCAADING STYLE SHEETS)	39
3.1 CSS SYNTAX.....	39
3.2 CSS SELECTORS	39
3.2.1 THE ELEMENT SELECTOR	39
3.2.2 THE ID SELECTOR	39

3.2.3	<i>THE CLASS SELECTOR</i>	40
3.2.4	<i>GROUPING SELECTORS</i>	40
3.3	WAYS TO INSERT CSS.....	41
3.3.1	<i>EXTERNAL STYLE SHEET</i>	41
3.3.2	<i>INTERNAL STYLE SHEET</i>	42
3.3.3	<i>INLINE STYLES</i>	42
3.3.4	<i>MULTIPLE STYLE SHEETS</i>	42
3.4	CSS BACKGROUND	43
3.4.1	<i>BACKGROUND COLOR</i>	43
3.4.2	<i>BACKGROUND IMAGE</i>	44
3.5	CSS TEXT.....	45
3.5.1	<i>TEXT COLOR</i>	45
3.5.2	<i>TEXT ALIGNMENT</i>	45
3.5.3	<i>TEXT DECORATION</i>	46
3.5.4	<i>TEXT TRANSFORMATION</i>	46
3.5.5	<i>TEXT INDENTATION</i>	47
3.6	CSS FONT.....	47
3.6.1	<i>FONT FAMILY</i>	47
3.6.2	<i>FONT STYLE</i>	47
3.6.3	<i>FONT SIZE</i>	48
3.7	CSS LINKS	49
3.8	CSS LISTS	49
3.8.1	<i>DIFFERENT LIST ITEM MARKERS</i>	49
3.8.2	<i>AN IMAGE AS THE LIST ITEM MARKER</i>	50
3.8.3	<i>CROSSBROWSER SOLUTION</i>	50
3.8.4	<i>LIST - SHORTHAND PROPERTY</i>	51
3.9	CSS TABLES.....	51
3.9.1	<i>COLLAPSE BORDERS</i>	51
3.9.2	<i>TABLE WIDTH AND HEIGHT</i>	52
3.9.3	<i>TABLE TEXT ALIGNMENT</i>	52
3.9.4	<i>TABLE PADDING</i>	52
3.9.5	<i>TABLE COLOR</i>	53
4	PHP	54
4.1	INSTALLATION OF A LOCAL SERVER.....	54
4.2	YOUR FIRST PHP PAGE.....	55
4.3	PHP BASICS	56
4.4	VARIABLES.....	58
4.5	CONDITIONAL STATEMENTS	59
4.5.1	<i>IF</i>	59
4.5.2	<i>IF ... ELSE</i>	59
4.5.3	<i>IF...ELSEIF...ELSE</i>	60
4.5.4	<i>SWITCH STATEMENT</i>	61
4.6	LOOPS.....	62
4.6.1	<i>THE WHILE LOOP</i>	62
4.6.2	<i>THE DO...WHILE STATEMENT</i>	63
4.6.3	<i>THE FOR LOOP</i>	64
4.6.4	<i>THE FOREACH LOOP</i>	65

4.7	PHP FUNCTIONS.....	66
4.7.1	PHP FUNCTIONS	66
4.7.2	CREATE A PHP FUNCTION.....	66
4.7.3	PHP FUNCTIONS – ADDING PARAMETERS	67
4.7.4	PHP FUNCTIONS - RETURN VALUES.....	68
4.8	ARRAYS.....	68
4.8.1	INDEXED ARRAYS	69
4.8.2	GET THE LENGTH OF AN ARRAY - THE COUNT() FUNCTION.....	70
4.8.3	LOOP THROUGH AN INDEXED ARRAY	70
4.8.4	ASSOCIATIVE ARRAYS	70
4.8.5	LOOP THROUGH AN ASSOCIATIVE ARRAY	71
4.9	PHP INCLUDE FILES	71
4.9.1	PHP INCLUDE AND REQUIRE STATEMENT.....	72
5	FORM HANDLING USING PHP.....	74
5.1	PHP \$_GET VARIABLE.....	74
5.2	PHP \$_POST FUNCTION	75
5.3	PHP \$_REQUEST FUNCTION	76
6	MYSQL.....	77
6.1	CREATE DATABASE.....	77
6.1.1	INTEGER VALUES	79
6.1.2	TEXT TYPES.....	80
6.1.3	CHAR.....	80
6.2	CONNECT TO THEMYSQL SERVER	82
6.2.1	OPEN A CONNECTION TO THE MYSQL SERVER.....	82
6.2.2	CLOSE A CONNECTION.....	83
6.3	CREATE DATABASE AND TABLES	83
6.3.1	CREATE A DATABASE.....	83
6.3.2	CREATE A TABLE.....	84
6.3.3	PRIMARY KEYS AND AUTO INCREMENT FIELDS.....	85
6.4	INSERT INTO.....	85
6.4.1	INSERT DATA INTO A DATABASE TABLE	85
6.4.2	INSERT DATA FROM A FORM INTO A DATABASE.....	86
6.5	SELECT	87
6.5.1	SELECT DATA FROM A DATABASE TABLE.....	87
6.5.2	DISPLAY THE RESULT IN AN HTML TABLE.....	88
6.6	THE WHERE CLAUSE	89
6.7	ORDER BY KEYWORD	90
6.8	THE UPDATE STATEMENT	91
6.9	THE DELETE STATEMENT.....	92
7	USER REGISTRATION AND AUTHENTICATION.....	94
7.1	USER REGISTRATION	94
7.1.1	HTML – REGISTRATION FORM.....	94
7.1.2	PHP SCRIPT FOR THE USER’S INFORMATION PROCESSING	96
7.1.3	STORING THE INFORMATION INTO DATABASE.....	97

7.1.4	PASSWORD ENCRYPTION.....	98
7.2	USER AUTHENTICATION.....	98
7.2.1	LOGIN FORM.....	98
7.2.2	AUTHENTICATION SCRIPT IN PHP.....	99
SUMMARY.....		105
THE LIST OF LITERATURE		106

INTRODUCTION

One of the subjects that is taught in the field of "Managerial Informatics", program of study "System Engineering and Informatics" at the Silesian University in Opava, School of Business Administration in Karviná is the subject of "Portal and its management A". This course as a free elective can be chosen by students of other fields of study. By monitoring of long-term interest, it is a popular subject for anyone interested in the issue of web pages respectively dynamic web pages and better web portals, which are now the primary communication interface in the Internet.

Current tools and technologies are increasingly closer to users and even people with non-informatics education are able to create a quality website that can be used for personal or professional purposes. And this is one of the primary objectives of the course "Portal and its management", in which students of field of study "Managerial Informatics" deepen your knowledge of dynamic websites and portals development and students of other disciplines, in our case related to economics, marketing, etc., penetrate into this issue and will get at least the basic knowledge and skills that they can, according to their own interests and needs, develop in the future.

In terms of content, subject is focused on promoting practical activities. From this perspective, main goal of this study textbook is presentation of web portal development using HTML (HyperText Markup Language (or XHTML (eXtensible HyperText Markup Language))), and CSS (Cascading Style Sheets), development of web forms, insertion of dynamic elements in the form of PHP (Hypertext Preprocessor) scripts, creation a database in MySQL, and basic methods of communication of web portals with databases MySQL (transmission of values between databases and web forms).

1 WEB & INTERNET PORTAL

If we start from the most general point of view, the term "portal" comes from the Latin porta (gate) and has a large number of meanings. Term "portal" can be used for example to describe architectural or other artistic features an ornate frame the entrance to major, usually public buildings such as the cathedral, exposed framing theatrical stage, which is suspended from a theater curtain, the entrance to a cave or other objects. In connection with the teaching of the subject "Portal and its management" we will discuss different meaning of this term in this text. We will address the issue called a web or internet portal. In this publication authors will, for reasons of uniformity and consistency of the text, use the term Web portal.

1.1 DEFINITION OF WEB PORTAL AND RELATED TERMS

The web portal can be defined from different perspectives based mainly on technologies, areas of application, target groups and functions (Definitions 1 – 3)

DEFINITION 1

As a portal in connection to information technology (IT) we consider web application that provides the user uniform and centralized information from various sources that the user is interested in or information relating to the user.

Another definition says that:

DEFINITION 2

Technically, the web portal is internet application connected to different data sources and working with him through an Internet browser.

In terms of functionality, usability and target groups may be further web portal defined as a:

DEFINITION 3

The web portal provides a secure and single point of interaction with diverse information, business processes and people adapted to the individual needs of particular users.

To further refine the definition of 1-3, it is necessary to define the term web applications (Definition 4) and following this definition even the concept of a Web server (Definition 5).

DEFINITION 4

In software engineering, web application is an application delivered to users from a web server over a computer network Internet, or its equivalent in-house (intranet).

DEFINITION 5

5 a) The web server is a computer that is responsible for handling HTTP requests from clients - program called a web browser.

5 b) The web server is a computer program that performs the activities described in the preceding paragraph (a demon).

Web portals are primarily based on client-server communication (Definition 6).

DEFINITION 6

The client-server is a network architecture, which separates the client (often an application with a graphical user interface), and a server who communicate over a computer network.

Client-server describes the relationship between two computer programs in which the first program, the client requests the services of another program called the server. An example is a Web browser, ie the client program on the user's computer that can access information at any web server in the world (the Internet) (of course with regard to authorize access).

1.2 TYPES OF WEB PORTALS

Web portals offer a wide range of services and information with possibilities of their customization according to personal needs and interests (personalization). In terms of services and focus on the target group, web portals are divided into three types, which are horizontal, vertical and combined.

1.2.1 HORIZONTAL WEB PORTALS

Horizontal web portals contain broad thematic focus and are oriented to the widest possible audience. This category includes traditional web search engines, for example:

- www.seznam.cz;
- www.google.com;
- www.centrum.cz;
- www.atlas.cz;
- www.volny.cz;
- www.yahoo.com;
- www.msn.com;
- www.aol.com;
- www.altavista.com;
- www.go.com;
- www.dogpile.com;
- and other.

1.2.2 VERTICAL WEB PORTALS

This type of web portals is aimed at a particular group of users (eg the community portals serve users in a particular region) or is oriented to a specific theme, again with the interests of certain groups of users. In terms of terminology this type of web portals is referred to as branch portals (vortals). As examples can be mentioned:

- www.diit.cz;
- www.statnisprava.cz;
- www.itnews.sk;
- management.blog.cz;
- www.microsoft.com;
- www.cd.cz;
- www.vodafone.cz;
- www.opf.slu.cz;
- www.idos.cz;
- and other.

Vertical web portals can be also specifically divided into the following types:

- Personal portals offering the option of personalizing the presentation and selection of information for its users (such as social networks, etc.);
- Regional portals;
- News portals;
- Government portals;
- Commercial and business portals;
- Enterprise portals (often used as an intranet);

1.2.3 COMBINATION OF A HORIZONTAL AND VERTICAL WEB PORTAL

As a special case of combined web portal we can present online stores that are oriented to different target groups (either a broad target group, or a narrow) and offer either specific products or a very wide range of products.

For vertical portals it is to be noted that the study text will not focus on the issue of enterprise portals, but only those which are created for the purpose of providing and sharing information on the Internet.

Just for clarification we state that enterprise portals are viewed as a web page (usually dynamic web pages) that is used to improve the publishing and sharing of information within the company. Basically, the terms enterprise, business, or internal portal is used to describe an application called intranet or extranet. Corporate portals are typically built on content management systems of type ECM (Enterprise Content Management) that can aggregate data from multiple data sources.

1.3 WEB PORTAL DEVELOPMENT

As result from the preceding text, web portals are in their concept of "dynamic web pages" presenting the user interface provides the user not only static content, but also a wide range of services, such as mail box, search, news in the mail, weather reports, discussion , chat, events calendar, catalogue and other sources. If we want to create a portal on their own or with help of an external supply, it is always necessary to conduct a detailed needs analysis based on a clear vision and strategy. Assumptions of the success of the project creating a web portal are:

- Identifying the needs of potential portal users;
- Specification of usability and user-friendliness;
- Technological options;
- Security;
- Stability;
- Reduction of technologies necessary for the functioning of the web portal;
- Maximizing flexibility portal for the needs of ongoing changes, improvements and additions;
- Timetable of portal development;
- Financial budget;
- The possibilities of web portal personalization;
- The testing of usability and user-friendliness.

In terms of development must be taken into account:

- Usability - a visitor has to easily find the information for which the web has come. Particularly the contacts, the demand or ordering goods must be after hands;
- Easy web site navigation - based on the above site usability. In particular the logical structure design and intuitive navigation for visitor;

- Web accessibility – it includes (not only) the technical requirements of such a significant and logical menu control of sites, alternative descriptions of images, font size, color references, etc.
- Valid code - search engines are now very sophisticated indexing algorithms that can cope with inappropriate code, but not the only reason to complicate them the work.

The basis is an orientation to the web site users. In seminars students will be create various projects in the form of web portals, respectively dynamic web pages. When you will create or manage web portal creation now or at any time later in practice, you should always follow some important principles that have become or are becoming standard and their applications in your semester projects can help you to try it in practice. In particular, the following list of basic principles relates to the above four areas:

- **Website content is accessible and readable:**
 - Each non-text element carrying information has a text alternative;
 - Information provided via scripts, objects, applets, cascading style sheets, images and other accessories to the user are available without any of these accessories;
 - Information provided by colors are also available without color resolution;
 - Foreground and background colors are sufficient contrast. The background is not a sample which reduces the readability;
 - Regulations specifying the font size do not use absolute units;
 - Determining the type of font contain general font family.
- **Work with the website is managed by the user:**
 - The web page content changes only when the user activates some elements;
 - Website without direct user command not manipulate the user interface;
 - New windows open only in justified cases and the user is warned in advance;
 - On the website, no blinking faster than once per second;
 - Website does not prevents the user from scrolling of frames;
 - Content or code of web pages does not even require a specific method of use or specific output or control devices.
- **Information is clear and understandable:**
 - Website information are presented using simple language and understandable form;
 - Home website clearly describes the meaning and purpose of the site. Name of the website or its operator is clear;
 - Website and individual elements of the text content present the key message at the beginning;
 - Extensive content blocks are divided into smaller, aptly titled units;
 - Information disclosed under the Act are available as text content of a webpage;
 - A separate web page includes contact to technical support and a statement clearly defining the accessibility level of the site and its parts. Each page of the site refers to this website;
- **Website navigation is clear and understandable:**
 - Each website has a meaningful name reflects its content;
 - Navigation and content information on the website are clearly separated;
 - Navigation is clear and consistent on all websites;
 - Each web page (except the homepage website) contains a link to a higher level in the hierarchy of the site and link to the homepage;
 - All websites contain extensive site links to a clear site map;
 - Content or code of web pages does not imply that the user has already visited another page;

- Each form element has a descriptive label assigned;
- Each frame has a suitable name or description which expresses its meaning and function.
- **Links are clear and provide guidance:**
 - The labeling of each link clearly describes its target without the surrounding context;
 - Just tagged links have the same goal;
 - Links are distinguished from other text, not only in color;
 - Side image maps are used only in the event that it was not possible with the available geometric shape defined areas in the image map. In other cases, image maps are used on the client side. An image map on the server side is always accompanied by alternative text links;
 - The user is clearly informed in advance when a link leads to a content type other than the website. Such a link is complemented by an information on the type and size of the target file;
- **The code is technically competent and structured:**
 - Web page code corresponds to published final specifications of HTML and XHTML. It does not contain syntax errors, which is the webmaster can remove;
 - In the meta tags is specified used character set of document;
 - The elements making headlines and lists are correctly marked in the source code; Elements which do not headings or lists, while in the source code are not marked;
 - For a description of the appearance of the website are the preferred style sheets;
 - If a table is used for layout of the web site does not contain header of rows or columns. All tables displaying tabular data however the header rows and / or columns contain;
 - All tables make sense when read line by line from left to right.

1.4 TOOLS AND TECHNOLOGIES FOR A WEB PORTAL DEVELOPMENT

For the development of a web portal it is possible to use a variety of technologies and tools we can use to creation of web pages with dynamic content. In terms of languages and scripting languages we can use HTML (HyperText Markup Language), XHTML (Extensible HyperText Markup Language), CSS (Cascading Style Sheets), Java Script, AJAX (Asynchronous JavaScript and XML (Extensible Markup Language)) PHP (Hypertext Preprocessor), JSP (JavaServer Pages), Java, Ruby, Python, Perl, ASP.NET, etc. They are now common languages that are mostly used in the latest versions, but in some cases the transition to a newer version usually does not cause any major problems.

In this textbook we will use:

- HTML or XHTML (the text using HTML with the fact that almost all codes correspond to the standard XHTML);
- Local server Apache;
- PHP;
- MySQL.

In terms of software support all the demos and examples will be implemented using PSPad and EasyPHP. In order to have right from the start interpreting test individual examples, in the preceding sentence above software PSPad will be first described. There is possible in this software without any supplements and other software, try the codes presented in Chapters HTML and CSS.

1.5 PSPAD

PSPad is a comprehensive text and code editor that provides not only advanced text editing functions, but also many tools addressed to developers, regardless of the programming language they use. The application has been specifically designed to serve a double purpose. It can easily replace the popular Wordpad and Notepad tools included in Windows and, in the same time, PSPad will prove its worth when dealing with complicated code syntax. The ace up its sleeve is actually a mix between the excellent bundle of features and its quite easy to use and friendly interface. Since it was designed to help all types of users, PSPad comes with an impressive list of templates that includes PHP, Python, HTML, C++, Foxpro, XHTML and many others. Here are some key features of "PSPad":

- work with projects;
- work on several documents at the same time (MDI);
- Save desktop session to later reopen all open files;
- FTP client - you can edit files directly from the web;
- macro recorder to record, save and load macros;
- search and replace in files;
- text difference with color-coded differences highlighted;
- templates (HTML tags, scripts, code templates...);
- installation contains templates for HTML, PHP, Pascal, JScript, VBScript, MySQL, MS-Dos, Perl,...;
- syntax highlighting auto set by file type;
- user-defined highlighters for exotic environments;
- auto correction;
- intelligent internal HTML preview using IE and Mozilla;
- full HEX editor;
- call external programs, different for each environment;
- external compiler with catch command output, log window, log parser for each environment for "IDE" effect;
- color syntax highlight printing and print preview;
- integrated TiDy library for formatting and checking HTML code, conversion to CSS, XML, XHTML;
- integrated free version of top CSS editor TopStyle Lite;
- export with highlight to RTF, HTML, TeX format into file or clipboard;
- column block select, bookmarks, line numbers, ...;
- reformat and compress HTML code, tags char case change;
- line sorting with ability to sort on defined column, with option to drop duplicates;
- ASCII chart with HTML entities;
- Code explorer for Pascal, INI, HTML, XML, PHP, and more in future;
- spell checker;
- internal web browser with APACHE support;
- matching bracket highlighting.

There is no complex installation process. PSPad is ready to work immediately without requiring customization. PSPad Software can be downloaded from a number of links. The basic link is <http://www.pspad.com/cz/>, but we can use, for example, <http://download.famouswhy.com/pspad/>. Installation steps are:

- 1) Download and run the .exe file from your computer.
- 2) Press "Next".

- 3) Read the license agreement and select "I accept the agreement". Press next to continue.
- 4) Select the installation folder and press "Next".
- 5) Select "Full installation" and press "Next".
- 6) Press "Install".
- 7) After the installation is complete, press "Finish" to exit setup.

2 HTML

2.1 CHARACTERISTICS OF HTML

Short for HyperText Markup Language, the authoring language used to create documents on the World Wide Web. HTML is similar to SGML (Standard Generalized Markup Language), although it is not a strict subset. Development of HTML was influenced by web browsers, which in turn back influenced the definition of the language.

2.2 CHARACTERISTICS OF XHTML

XHTML is almost identical to HTML 4.01 with only few differences. This is a cleaner and more strict version of HTML 4.01. If you already know HTML then you need to give little attention to learn this latest variant of HTML. XHTML stands for EXtensible HyperText Markup Language and is the next step in the evolution of the Internet. The XHTML 1.0 is the first document type in the XHTML family. XHTML was developed by the W3C to help web developers make the transition from HTML to XML. By migrating to XHTML today, web developers can enter the XML world with all of its attendant benefits, while still remaining confident in their content's backward and future compatibility. Developers who migrate their content to XHTML 1.0 will realize the following benefits:

- XHTML documents are XML conforming. As such, they are readily viewed, edited, and validated with standard XML tools.
- XHTML documents can be written to operate better than they did before in existing browsers as well as in new browsers.
- XHTML documents can utilize applications like scripts and applets that rely upon either the HTML Document Object Model or the XML Document Object Model.

2.3 FUNDAMENTAL DIFFERENCES BETWEEN HTML AND XHTML

XML syntax rules are far more rigorous than HTML. As a result, XHTML makes authors work more precisely, having to address issues such as:

- all elements and attribute names must appear in lower case;
- all attribute values must be quoted;
- non-Empty Elements require a closing tag;
- empty elements are terminated using a space and a trailing slash;
- no attribute minimization is allowed;
- in strict XHTML, all inline elements must be contained in a block element.

In HTML, case, quotes, termination of many elements and uncontained elements are allowed and commonplace. The margin for errors in HTML is much broader than in XHTML, where the rules are very clear. As a result, XHTML is easier to author and to maintain, since the structure is more apparent and problem syntax is easier to spot.

2.4 HTML TAGS AND ELEMENTS

Every tag consists of a tag name, sometimes followed by an optional list of tag attributes, all placed between opening and closing brackets (< and >). The simplest tag is nothing more than a name appropriately enclosed in brackets, such as <head> and <i>.

More complicated tags contain one or more attributes, which specify or modify the behavior of the tag.

EXAMPLE 1

General notation:

```
<tagname>content</tagname>
```

Specific example:

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

According to the HTML standard, tag and attribute names are not case-sensitive. There's no difference in effect between `<head>`, `<Head>`, `<HEAD>`, or even `<HeaD>`; they are all equivalent. But with XHTML, case is important: all current standard tag and attribute names are in lowercase.

The purpose of a web browser (such as Google Chrome, Internet Explorer, Firefox, Safari) is to read HTML documents and display them as web pages. "HTML tags" and "HTML elements" are often used to describe the same thing. But strictly speaking, an HTML element is everything between the start tag and the end tag, including the tags.

A very good quality associated with all the browsers is that they would not give any error if you have not put any HTML tag or attribute properly. They will just ignore that tag or attribute and will apply only correct tags and attributes before displaying the result.

2.5 HTML DOCUMENT

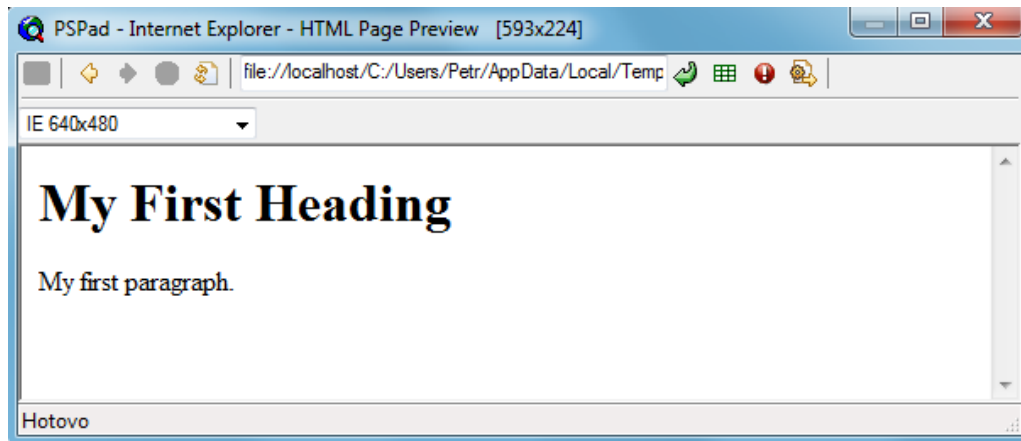
To begin coding HTML you need only two things: a simple editor (we will use PsPad) and a web browser.

The DOCTYPE declaration defines the document type. The first line on the top, `<!DOCTYPE html>`, is a document type declaration and it lets the browser know which flavor of HTML we are using. `<html>` is the opening tag that kicks things off and tells the browser that everything between that and the `</html>` closing tag is an HTML document. The stuff between `<body>` and `</body>` is the main content of the document that will appear in the browser window. The `</body>` and `</html>` put a close to their respective elements.

EXAMPLE 2

```
<!DOCTYPE html>
<html>
  <body>
    <h1>My First Heading</h1>

    <p>My first paragraph.</p>
  </body>
</html>
```

The `<!DOCTYPE>` declaration must be the very first thing in your HTML document, before the `<html>` tag. The `<!DOCTYPE>` declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in. In HTML 4.01, the `<!DOCTYPE>` declaration refers to a DTD, because HTML 4.01 was based on SGML (Standard Generalized Markup Language). The DTD specifies the rules for the markup language, so that the browsers render the content correctly. HTML5 is not based on SGML, and therefore does not require a reference to a DTD.

2.6 HTML ELEMENT SYNTAX

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

Not all tags have closing tags like this (`<html></html>`) some tags, which do not wrap around content will close themselves. The line-break tag for example, looks like this: `
` - a line break doesn't hold any content so the tag merrily sits by its lonely self.

In Example 2 there are two basic tags. The elements `h1`, `h2`, `h3`, `h4`, `h5` and `h6` is used to make headings (h stands for "heading"), where `h1` is the first level and normally the largest text, `h2` is the second level and normally slightly smaller text, and `h6` is the sixth and last in the hierarchy of headings and normally the smallest text.

The `p` in `<p>` is short for "paragraph" which is exactly what it is - a text paragraph.

2.7 HTML PARAGRAPHS AND LINE BREAKS

Paragraphs are defined with the `<p>` tag.

EXAMPLE 3

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

Note that browsers automatically add an empty line before and after a paragraph. Most browsers will display HTML correctly even if you forget the end tag:

EXAMPLE 4

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

The example 4 will work in most browsers, but don't rely on it. Forgetting the end tag can produce unexpected results or errors. Future version of HTML will not allow you to skip end tags.

If you want a line break (a new line) without starting a new paragraph, use the `
` tag. The `
` element is an empty HTML element. It has no end tag.

EXAMPLE 5

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

Just as `
` we can use tag `<hr>`. The `<hr>` tag creates a horizontal line in an HTML page.

EXAMPLE 6

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

2.8 HTML HEADINGS

As was already indicated in Example 2Headings are defined with the `<h1>` to `<h6>` tags. `<h1>` defines the most important heading. `<h6>` defines the least important heading.

EXAMPLE 7

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

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

2.9 HTML FONT

The `` tag provides no real functionality by itself, but with the help of a few attributes, this tag is used to change the style, size, and color of HTML text elements. The

size, color, and face attributes can be used all at once or individually, providing users with the ability to create dynamic font styles for any HTML element.

You can set the size of your font with *size*. The range of accepted values goes from 1 -- the smallest, to 7 -- the largest. The default *size* of a font is 3.

EXAMPLE 8

```
<p>
<font size="5">Here is a size 5 font</font>
</p>
```

Color of your font can be set with *color*.

EXAMPLE 9

```
<font color="#990000">This text is hex color #990000</font>
<br />
<font color="red">This text is red</font>
```

Basic color codes can be found in Table 1. Much more may be found, for example, in:

- <http://www.computerhope.com/htmcolor.htm>;
- <http://www.december.com/html/spec/colorcodes.html>;
- <http://www.htmlgoodies.com/tutorials/colors/article.php/3478961>.

Table 1: Basic colors codes

Black	Navy	Blue	Green	Teal	Lime	Aqua	Maroon
#000000	#000080	#0000ff	#009900	#009999	#00ff00	#00ffff	#800000
							
Purple	Olive	Gray	Silver	Red	Fuchsia	Yellow	White
#800080	#999900	#808080	#cccccc	#ff0000	#ff00ff	#ffff00	#ffffff
							

A different font face can be chosen by specifying any font you have installed. Font face is synonymous with font type. As a web designer, be aware that if you specify a custom font type and users viewing the page don't have the exact same font installed, they will not be able to see it. Instead the chosen font will default to Times New Roman. To reduce the risk of running into this situation, you may provide a list of several fonts with the face attribute, such as outlined below.

EXAMPLE 10

```
<p>
<font face="Georgia, Arial, Garamond">This paragraph
has had its font...</font>
</p>
```

In general, the `` and `<basefont>` tags are deprecated and should not be used for published work. Instead, use CSS (Cascading style sheets) styles to manipulate your font. CSS will be described in the next section.

2.10 HTML LISTS

HTML lists appear in web browsers as bulleted lines of text. There are actually three different types of HTML lists, including unordered lists (bullets), ordered lists (numbers), and definition lists (think: dictionaries). Each list type utilizes its own unique list tag.

EXAMPLE 11

```
<body>
  <ul>
    <li>I am a list item!>
    <li>I am a list item too!>
    <li>I am a list item also!>
  </ul>
</body>
```

The actual list tags themselves, such as ``, are nothing but container elements for list items (``). They work behind the scenes to identify the beginning and ending of an HTML list element.

An unordered list (``) signifies to a web browser that all list items contained inside the `` tag should be rendered with a bullet preceding the text. The default bullet type for most web browsers is a full disc (black circle), but this can be adjusted using an HTML attribute called `type`.

EXAMPLE 12

```
<h4 align="center">Shopping List</h4>

<ul>
  <li>Milk</li>
  <li>Toilet Paper</li>
  <li>Cereal</li>
  <li>Bread</li>
</ul>
```

To render a list with a different bullet type, add a `type` attribute to the unordered list element. Using the same code in the example above, replace the `` line from the previous example with any of the lines listed below to change the bullet from disc shape to another shape.

EXAMPLE 13

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

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

An ordered list is defined using the `` tag, and list items placed inside of an ordered list are preceded with numbers instead of bullets.

EXAMPLE 14

```
<h4 align="center">Goals</h4>

<ol>
  <li>Find a Job</li>
  <li>Get Money</li>
  <li>Move Out</li>
</ol>
```

The numbering of an HTML list can be changed to letters or Roman Numerals by once again adjusting the type attribute.

EXAMPLE 15

```
<ol type="a">
<ol type="A">
<ol type="i">
<ol type="I">
```

The start attribute allows you to further customize an HTML ordered list by setting a new starting digit for the ordered list element.

EXAMPLE 16

```
<h4 align="center">Goals</h4>

<ol start="4" >
  <li>Buy Food</li>
  <li>Enroll in College</li>
  <li>Get a Degree</li>
</ol>
```

HTML definition lists (`<dl>`) are list elements that have a unique array of tags and elements; the resulting listings are similar to those you'd see in a dictionary.

- `<dl>` - opening clause that defines the start of the list;
- `<dt>` - list item that defines the definition term;
- `<dd>` - definition of the list item.

These lists displace the word term (`<dt>`) in such a way that it rests just above the definition element (`<dd>`) to offer a very unique look. For best results we suggest bolding the definition terms with the bold tag ``.

EXAMPLE 17

```
<dl>
```

```

<dt><b>Fromage</b></dt>
  <dd>French word for cheese.</dd>
<dt><b>Voiture</b></dt>
  <dd>French word for car.</dd>
</dt>
</dl>

```

2.11 HTML IMAGES

In HTML, images are defined with the `` tag. The `` tag is empty, which means that it contains attributes only, and has no closing tag. To display an image on a page, you need to use the `src` attribute. `src` stands for "source". The value of the `src` attribute is the URL of the image you want to display.

EXAMPLE 18

```

```

The URL points to the location where the image is stored. For example, an image named `boat.gif`, located in the `images` directory on `www.w3schools.com` has the URL: <http://www.w3schools.com/images/boat.gif>. The browser displays the image where the `` tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

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

EXAMPLE 19

```

```

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

The `height` and `width` attributes are used to specify the height and width of an image. The attribute values are specified in pixels by default:

EXAMPLE 20

```

```

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

Above, we've used hard-coded pixel values for the `height` and `width` of the sunset image to ensure that this image will always render 42 pixels high by 42 pixels wide. By hard-coding these values, we are ensuring that the image will only display 42 pixels high by 42 pixels wide, even if the picture file itself happens to be much larger. If the dimensions of the

picture are much larger, then we risk some severe skewing as the browser tries to shrink our image into our small box.

Height and width values can also be a percentage. Percentage values are relative to the parent HTML element (usually the body), which means if you have a parent element like a `<body>` element that is the whole screen (1024x768), then an image with a height and width of 100% will take up that entire body element (1024x768). In our example below, we have placed the image in a table element that is about 400 pixels wide by 200 pixels tall.

Images can be aligned horizontally to the right or to the left of other elements using the `align` attribute. Image elements are aligned to the left by default.

EXAMPLE 21

```

```

As you can see, the image's right edge has now been aligned with the right edge of the display box. Since the display box is the parent element, this is the desired behavior for the `align` attribute. If we take this example a step further, you can achieve some really great designs by embedding aligned images inside of paragraph `<p>` elements.

EXAMPLE 22

```
<p>This is paragraph 1, yes it is...</p>
<p>

The image will appear along the...isn't it?
</p>
<p>This is the third paragraph that appears...</p>
```

2.12 HTML TABLES

Tables are defined with the `<table>` tag. A table is divided into rows with the `<tr>` tag (tr stands for table row). A row is divided into data cells with the `<td>` tag (td stands for table data). A row can also be divided into headings with the `<th>` tag (th stands for table heading). The `<td>` elements are the data containers in the table. The `<td>` elements can contain all sorts of HTML elements like text, images, lists, other tables, etc.

EXAMPLE 23

```
<table>
  <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>
```

If you do not specify a border for the table, it will be displayed without borders. A border can be added using the border attribute:

EXAMPLE 24

```
<table border="1">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Table headings are defined with the <th> tag. By default, all major browsers display table headings as bold and centered:

EXAMPLE 25

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Points</th>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Content elements like HTML lists, images, and even other table elements can be placed inside each table cell. Doing so aligns the elements in a tabular fashion and provides structure.

EXAMPLE 26

```
<table border="1">
  <tr>
    <td width="50%">
      <ul>
        <li>List Item 1</li>
        <li>List Item 2</li>
        <li>List Item 3</li>
      </ul>
    </td>
    <td>
      <ul>
```



```

        <li>List Item 4</li>
        <li>List Item 5</li>
        <li>List Item 6</li>
    </ul>
</td>
</tr>
<tr>
    <td>
        <p>Avoid losing floppy disks with important
school...</p>
    </td>
    <td>
        <a href="http://www.espn.com" target="_blank"
rel="nofollow">
            
        </a>
    </td>
</tr>
</table>

```

HTML tables allow the web designer to align page content in a tabular fashion while spanning elements horizontally across the web page, rather than stacking them up one on top of another.

A table can contain an infinite number of table rows. Each table row is essentially a table element itself, with an opening and closing tag (<tr> </tr>). Table columns are also considered child elements of HTML tables, and like table rows, an HTML table may contain an infinite number of table data cells (<td> <tr>).

Table rows and columns are container elements that house other HTML elements such as text links, images, and lists, as we've seen in previous examples. Below, we've applied a background color to the table example in order to help distinguish the different table elements.

EXAMPLE 27

```

<table border="1">
    <tr title="You are looking at Row 1" bgcolor="silver">
        <td>Row 1 Cell 1</td>
        <td>Row 1 Cell 2</td>
    </tr>
    <tr title="You are looking at Row 2" bgcolor="aqua">
        <td>Row 2 Cell 1</td>
        <td>Row 2 Cell 2</td>
    </tr>
</table>

```

You can use `rowspan` to span multiple rows merging together table rows and `colspan` to span across multiple columns.

EXAMPLE 28

```

<table border="1">

```

```

<tr>
  <td><b>Column 1</b></td>
  <td><b>Column 2</b></td>
  <td><b>Column 3</b></td>
</tr>
<tr>
  <td rowspan="2">Row 1 Cell 1</td>
  <td>Row 1 Cell 2</td>
  <td>Row 1 Cell 3</td>
</tr>
<tr>
  <td>Row 2 Cell 2</td>
  <td>Row 2 Cell 3</td>
</tr>
<tr>
  <td colspan="3">Row 3 Cell 1</td>
</tr>
</table>

```

With the `cellpadding` and `cellspacing` attributes, you will be able to adjust the spacing between table cells. Setting the `cellpadding` attribute determines how much space will exist between a table cell border and the elements contained within it, whereas `cellspacing` determines how much space will exist between each table cell. Color has been added to the table below to emphasize these attributes.

EXAMPLE 29

```

<table border="1" cellspacing="10" bgcolor="rgb(0,255,0)">
  <tr>
    <td><b>Column 1</b></td>
    <td><b>Column 2</b></td>
  </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>

```

And now we will change the `cellpadding` of the table and remove the `cellspacing` from the previous example. This should clearly demonstrate the difference between `cellpadding` and `cellspacing`.

EXAMPLE 30

```

<table border="1" cellpadding="10" bgcolor="rgb(0,255,0)">
  <tr>
    <td><b>Column 1</b></td>
  </tr>
</table>

```

```

        <td><b>Column 2</b></td>
    </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>

```

2.13 HTML LINKS

The HTML `<a>` tag defines a hyperlink. A hyperlink (or link) is a word, group of words, or image that you can click on to jump to another document. When you move the cursor over a link in a Web page, the arrow will turn into a little hand. The most important attribute of the `<a>` element is the `href` attribute, which indicates the link's destination. By default, links will appear as follows in all browsers:

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

EXAMPLE 31

```
<a href="url">Link text</a>
```

The `href` attribute specifies the destination of a link.

EXAMPLE 32

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

The `target` attribute specifies where to open the linked document. The example below will open the linked document in a new browser window or a new tab:

EXAMPLE 33

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

The `id` attribute can be used to create a bookmark inside an HTML document. An anchor with an `id` inside an HTML document:

EXAMPLE 34

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

Create a link to the "Useful Tips Section" inside the same document:

EXAMPLE 35

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

Or, create a link to the "Useful Tips Section" from another page:

EXAMPLE 36

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

Placing files available for download is done in exactly the same fashion as placing text links. However, things become complicated if we want to place image links for download.

EXAMPLE 37

```
<a href="http://www.tizag.com/pics/htmlT/blanktext.zip">Text
Document</a>
```

We can also use image links. Image links are constructed as you might expect, by embedding an `` tag inside of an anchor element `<a>`. Like HTML text links, image links require opening and closing anchor tags, but instead of placing text between these opening and closing tags, the developer needs to place an image tag -- with a valid source attribute value of course.

EXAMPLE 38

```
<a href="http://www.espn.com" target="_blank">
  
</a>
```

By default, many browsers add a small border around image links. This default behavior is intended to give web viewers the ability to quickly decipher the difference between ordinary images and image links. Since this default is different from web browser to web browser, it may be best to squelch this ambiguity and set the border attribute of the image tag to 0 or 1.

EXAMPLE 39

```
<a href="http://www.espn.com" target="_blank">
  
</a>
```

Thumbnails are by far the most common type of image link seen in today's world. To create a thumbnail, one must save a low-quality version of a picture with smaller dimensions. Then, one should link this low-quality picture to its higher-quality counterpart. Thumbnails are intended to give your audience quick previews of images without them having to wait for the larger, higher-quality image to load. Photo galleries make heavy use of thumbnails, and they will allow you to display multiple pictures on one page with ease.

EXAMPLE 40

```
<a href="sunset.gif">
  
</a>
```

Use the `<base>` tag in the head element to set a default URL for all links on a page to go to. It's always a good idea to set a base tag just in case your links become bugged somewhere down the line. Usually, you should set your base to your home page.

EXAMPLE 41

```
<head>
  <base href="http://www.tizag.com/" />
</head>
```

2.14 FRAMES

Frames divide a browser window into several pieces or panes, each pane containing a separate XHTML/HTML document. One of the key advantages that frames offer is that you can then load and reload single panes without having to reload the entire contents of the browser window. A collection of frames in the browser window is known as a frameset. The window is divided up into frames in a similar pattern to the way tables are organized: into rows and columns. The simplest of framesets might just divide the screen into two rows, while a complex frameset could use several rows and columns. There are few drawbacks also you should be aware of with frames are as follows:

- Some browsers do not print well from framesets.
- Some smaller devices cannot cope with frames, often because their screen is not big enough to be divided up.
- Some time 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 who do not support frame technology.

To create a frameset document, first you need the `<frameset>` element, which is used instead of the `<body>` element. The frameset defines the rows and columns your page is divided into, which in turn specify where each individual frame will go. Each frame is then represented by a `<frame>` element.

You also need to learn the `<noframes>` element, which provides a message for users whose browsers do not support frames.

2.14.1 CREATING FRAMES

- The `<frameset>` tag replaces the `<body>` element in frameset documents.
- The `<frameset>` tag defines how to divide the window into frames.
- Each frameset defines a set of rows or columns. If you define frames by using rows then horizontal frames are created. If you define frames by using columns then vertical frames are created.
- The values of the rows/columns indicate the amount of screen area each row/column will occupy.
- Each frame is indicated by `<frame>` tag and it defines what HTML document to put into the frame.

Following is the example to create three horizontal frames:

EXAMPLE 42

```

<html>
<head>
<title>Frames example</title>
</head>
  <frameset rows="10%,80%,10%">
    <frame src="/html/top_frame.htm" />
    <frame src="/html/main_frame.htm" />
    <frame src="/html/bottom_frame.htm" />
  <noframes>
  <body>
  Your browser does not support frames.
  </body>
  </noframes>
</frameset>
</html>

```

2.14.2 THE <FRAMESET> ELEMENT ATTRIBUTES

Following are important attributes of <frameset> and should be known to you to use frameset.

- **cols:** 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 four ways:
 - Absolute values in pixels. For example to create three vertical frames, use cols="100, 500,100".
 - A percentage of the browser window. For example to create three vertical frames, use cols="10%, 80%,10%".
 - Using a wildcard symbol. For example to create three vertical frames, use cols="10%, *,10%". In this case wildcard takes remainder of the window.
 - As relative widths of the browser window. For example to create three vertical frames, use cols="3*,2*,1*". 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.
- **rows:** attribute works just like the cols attribute and can take the same values, but it is used to specify the rows in the frameset. For example to create two horizontal frames, use rows="10%, 90%". You can specify the height of each row in the same way as explained above for columns.
- **border:** attribute specifies the width of the border of each frame in pixels. For example border="5". A value of zero specifies that no border should be there.
- **frameborder:** specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder="0" specifies no border.
- **framespacing:** specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing="10" means there should be 10 pixels spacing between each frames.

2.14.3 LOADING CONTENT

The `<frame>` element indicates what goes in each frame of the frameset. The `<frame>` element is always an empty element, and therefore should not have any content, although each `<frame>` element should always carry one attribute, `src`, to indicate the page that should represent that frame. From the above example, let's take a small snippet:

EXAMPLE 43

```
<frame src="/html/top_frame.htm" />
  <frame src="/html/main_frame.htm" />
  <frame src="/html/bottom_frame.htm" />
```

2.14.4 THE `<FRAME>` ELEMENT ATTRIBUTES

Following are important attributes of and should be known to you to use frames.

- **src**: indicates the file that should be used 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**: 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 a second frame, in which case the second frame needs a name to identify itself as the target of the link.
- **frameborder**: attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the `frameborder` attribute on the `<frameset>` element if one is given, and the possible values are the same. This can take values either 1 (yes) or 0 (no).
- **marginwidth**: 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**: 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**: 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**: allows you to provide a link to another page containing a long description of the contents of the frame. For example `longdesc="framedescription.htm"`.

2.14.5 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 the pages with the content into a separate frame.

As you have already seen, each `<frame>` element can carry the `name` attribute to give each frame a name. This name is used in the links to indicate which frame the new page

should load into. Consider this very simple example, create following content in index.htm file:

EXAMPLE 44

```
<frameset cols="200, *">
  <frame src="/html/menu.htm" name="menu_page" />
  <frame src="/html/main.htm" name="main_page" />
</frameset>
```

There are two columns in this example. The first is 200 pixels wide and will contain the navigation bar. The second column or frame will contain the main part of the page. The links on the left side navigation bar will load pages into the right side main page. Keep some content in main.htm file and the links in the menu.htm file look like this:

EXAMPLE 45

```
<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>
```

2.15 FORMS

HTML Forms are required when you want to collect some data from the site visitor. For example registration information: name, email address, credit card, etc. A form will take input from the site visitor and then will post your back-end application such as CGI (Common Gateway Interface), ASP (Active Server Pages) Script or PHP (Hypertext Preprocessor) script etc. Then your back-end application will do required processing on that data in whatever way you like. Form elements are like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc. which are used to take information from the user. A simple syntax of using `<form>` is as follows:

EXAMPLE 46

```
<form action="back-end script" method="posting method">
  form elements like input, textarea etc.
</form>
```

Most frequently used form attributes are:

- **name:** This is the name of the form.
- **action:** Here you will specify any script URL which will receive uploaded data.
- **method:** Here you will specify method to be used to upload data. It can take various values but most frequently used are GET and POST.
- **target:** It specifies the target page 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 like:
 - application/x-www-form-urlencoded - This is the standard method most forms use. It converts spaces to the plus sign and non-alphanumeric characters into the hexadecimal code for that character in ASCII text.
 - multipart/form-data - This allows the data to be sent in parts, with each consecutive part corresponding to a form control, in the order they appear in the form. Each part can have an optional content-type header of its own indicating the type of data for that form control.

There are different types of form controls that you can use to collect data from a visitor to your site.

- Text input controls;
- Buttons;
- Checkboxes and radio buttons;
- Select boxes;
- File select boxes;
- Hidden controls;
- Submit and reset button.

2.15.1 TEXT INPUT CONTROLS

There are actually three types of text input used on forms:

- **Single-line text input controls:** Used for items that require only one line of user input, such as search boxes or names. They are created using the `<input>` element.
- **Password input controls:** Single-line text input that mask the characters a user enters.
- **Multi-line text input controls:** Used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created with the `<textarea>` element.

2.15.2 SINGLE-LINE TEXT INPUT CONTROLS

Single-line text input controls are created using an `<input>` element whose type attribute has a value of text. Here is a basic example of a single-line text input used to take first name and last name:

EXAMPLE 47

```
<form action="/cgi-bin/hello_get.cgi" method="get">
First name:
<input type="text" name="first_name" />
<br>
Last name:
<input type="text" name="last_name" />
<input type="submit" value="submit" />
</form>
```

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

- **type:** Indicates the type of input control you want to create. This element is also used to create other form controls such as radio buttons and checkboxes.

- **name:** Used to give the name part of the name/value pair that is sent to the server, representing each form control and the value the user entered.
- **value:** Provides an initial value for the text input control that the user will see when the form loads.
- **size:** Allows you to specify the width of the text-input control in terms of characters.
- **maxlength:** Allows you to specify the maximum number of characters a user can enter into the text box.

2.15.3 PASSWORD INPUT CONTROLS

This is also a form of single-line text input controls are created using an `<input>` element whose type attribute has a value of password. Here is a basic example of a single-line password input used to take user password:

EXAMPLE 48

```
<form action="/cgi-bin/hello_get.cgi" method="get">
Login :
<input type="text" name="login" />
<br>
Password:
<input type="text" name="password" />
<input type="submit" value="submit" />
</form>
```

2.15.4 MULTIPLE-LINE TEXT INPUT CONTROLS

If you want to allow a visitor to your site to enter more than one line of text, you should create a multiple-line text input control using the `<textarea>` element. Here is a basic example of a multi-line text input used to take item description:

EXAMPLE 49

```
<form action="/cgi-bin/hello_get.cgi" method="get">
Description : <br />
<textarea rows="5" cols="50" name="description">
Enter description here...
</textarea>
<input type="submit" value="submit" />
</form>
```

Following is the detail of above used attributes for `<textarea>` tag.

- **name:** The name of the control. This is used in the name/value pair that is sent to the server.
- **rows:** Indicates the number of rows of text area box.
- **cols:** Indicates the number of columns of text area box.

2.15.5 CREATING BUTTON

There are various ways in HTML to create clickable buttons. You can create clickable button using `<input>` tag. When you use the `<input>` element to create a button, the type of button you create is specified using the type attribute. The type attribute can take the following values:

- **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.

EXAMPLE 50

```
<form action="http://www.example.com/test.asp" method="get">
  <input type="submit" name="Submit" value="Submit" />
  <br /><br />
  <input type="reset" value="Reset" />
  <input type="button" value="Button" />
</form>
```

You can use an image to create a button.

EXAMPLE 51

```
<form action="http://www.example.com/test.asp" method="get">
  <input type="image" name="imagebutton" src="URL" />
</form>
```

You can use `<button>` element to create various buttons.

EXAMPLE 52

```
<form action="http://www.example.com/test.asp" method="get">
  <button type="submit">Submit</button>
  <br /><br />
  <button type="reset"> Reset </button>
  <button type="button"> Button </button>
</form>
```

2.15.6 CHECKBOXES CONTROL

Checkboxes are used when more than one option is required to be selected. They are created using `<input>` tag as shown below. Here is example HTML code for a form with two checkboxes:

EXAMPLE 53

```
<form action="/cgi-bin/checkbox.cgi" method="get">
  <input type="checkbox" name="maths" value="on"> Maths
  <input type="checkbox" name="physics" value="on"> Physics
  <input type="submit" value="Select Subject" />
```

```
</form>
```

Following is the list of important checkbox attributes:

- **type**: Indicates that you want to create a checkbox.
- **name**: Name of the control.
- **value**: The value that will be used if the checkbox is selected. More than one checkbox should share the same name only if you want to allow users to select several items from the same list.
- **checked**: Indicates that when the page loads, the checkbox should be selected.

2.15.7 RADIO BOX CONTROL

Radio Buttons are used when only one option is required to be selected. They are created using `<input>` tag as shown below. Here is example HTML code for a form with two radio button:

EXAMPLE 54

```
<form action="/cgi-bin/radiobutton.cgi" method="post">
<input type="radio" name="subject" value="maths" /> Maths
<input type="radio" name="subject" value="physics" /> Physics
<input type="submit" value="Select Subject" />
</form>
```

Following is the list of important radiobox attributes:

- **type**: Indicates that you want to create a radiobox.
- **name**: Name of the control.
- **value**: Used to indicate the value that will be sent to the server if this option is selected.
- **checked**: Indicates that this option should be selected by default when the page loads.

2.15.8 SELECT BOX CONTROL

Drop Down Box is used when we have many options available to be selected but only one or two will be selected. Here is example HTML code for a form with one drop down box:

EXAMPLE 55

```
<form action="/cgi-bin/dropdown.cgi" method="post">
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
<input type="submit" value="Submit" />
</form>
```

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

- **name**: This is the name for the control.
- **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>:

- **value:** The value that is sent to the server if this option is selected.
- **selected:** Specifies that this option should be the initially selected value when the page loads.
- **label:** An alternative way of labeling options.

2.15.9 FILE SELECT BOXES

If you want to allow a user to upload a file to your web site from his computer, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element. Here is example HTML code for a form with one file select box:

EXAMPLE 56

```
<form action="/cgi-bin/hello_get.cgi" method="post"
      name="fileupload" enctype="multipart/form-data">
<input type="file" name="fileupload" accept="image/*" />
</form>
```

2.15.10 HIDDEN CONTROLS

If you want to pass information between pages without the user seeing it. Hidden form controls remain part of any form, but the user cannot see them in the Web browser. They should not be used for any sensitive information you do not want the user to see because the user could see this data if she looked in the source of the page. Following hidden form is being used to keep current page number. When a user will click next page then the value of hidden form will be sent to the back-end application and it will decide which page has be displayed next.

EXAMPLE 57

```
<form action="/cgi-bin/hello_get.cgi"
      method="get" name="pages">
<p>This is page 10</p>
<input type="hidden" name="page number" value="10" />
<input type="submit" value="Next Page" />
</form>
```

2.15.11 SUBMIT AND RESET BUTTON

These are special buttons which can be created using <input> When submit button is clicked then Forms data is submitted to the back-end application. When reset button is clicked then all the forms control are reset to default state. You already have seen submit button above, we will give one reset example here:

EXAMPLE 58

```
<form action="/cgi-bin/hello_get.cgi" method="get">
First name:
```

2HTML

```
<input type="text" name="first_name" />  
<br>  
Last name:  
<input type="text" name="last_name" />  
<input type="submit" value="Submit" />  
<input type="reset" value="Reset" />  
</form>
```

3 CSS (CASCADING STYLE SHEETS)

HTML was never intended to contain tags for formatting a document. HTML was intended to define the content of a document, like: `<h1>This is a heading</h1>`, `<p>This is a paragraph.</p>`. When tags like ``, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large web sites, where fonts and color information were added to every single page, became a long and expensive process. To solve this problem, the World Wide Web Consortium (W3C) created CSS.

3.1 CSS SYNTAX

A CSS rule set consists of a selector and a declaration block:

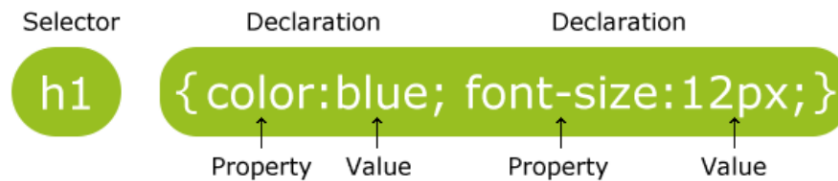


Figure 1: CSS Syntax

Source: http://www.w3schools.com/css/css_syntax.asp

The selector points to the HTML element you want to style. The declaration block contains one or more declarations separated by semicolons. Each declaration includes a property name and a value, separated by a colon.

3.2 CSS SELECTORS

CSS selectors allow you to select and manipulate HTML element(s). CSS selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much more.

3.2.1 THE ELEMENT SELECTOR

The element selector selects elements based on the element name. You can select all `<p>` elements on a page like this: (all `<p>` elements will be center-aligned, with a red text color).

EXAMPLE 59

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

3.2.2 THE ID SELECTOR

The id selector uses the id attribute of an HTML tag to find the specific element. An `id` should be unique within a page, so you should use the id selector when you want to find a single, unique element. To find an element with a specific id, write a hash character,

followed by the id of the element. The style rule below will be applied to the HTML element with `id="para1"`:

EXAMPLE 60

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

3.2.3 THE CLASS SELECTOR

The class selector finds elements with the specific class. The `class` selector uses the HTML class attribute. To find elements with a specific class, write a period character, followed by the name of the class. In the example below, all HTML elements with `class="center"` will be center-aligned:

EXAMPLE 61

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

You can also specify that only specific HTML elements should be affected by a class. In the example below, all `p` elements with `class="center"` will be center-aligned:

EXAMPLE 62

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

3.2.4 GROUPING SELECTORS

In style sheets there are often elements with the same style:

EXAMPLE 63

```
h1 {
    text-align: center;
    color: red;
}

h2 {
    text-align: center;
    color: red;
}
```



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

To minimize the code, you can group selectors. To group selectors, separate each selector with a comma. In the example below we have grouped the selectors from the code above:

EXAMPLE 64

```
h1, h2, p {
    text-align: center;
    color: red;
}
```

3.3 WAYS TO INSERT CSS

There are three ways of inserting a style sheet:

- External style sheet;
- Internal style sheet;
- Inline style.

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 just one file. Each page must include a link to the style sheet with the <link> tag. The <link> tag goes inside the head section:

EXAMPLE 65

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

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

EXAMPLE 66

```
"myStyle.css":

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

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 of an HTML page, inside the `<style>` tag, like this:

EXAMPLE 67

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

3.3.3 INLINE STYLES

An inline style loses many of the advantages of a style sheet (by mixing content with presentation). Use this method sparingly! To use inline styles, add the style attribute to 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:

EXAMPLE 68

```
<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, assume that an external style sheet has the following properties for the h3 selector:

EXAMPLE 69

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

then, assume that an internal style sheet also has the following properties for the h3 selector:

EXAMPLE 70

```
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 the h3 element will be:

EXAMPLE 71

```
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

CSS background properties are used to define the background effects of an element. CSS properties used for background effects:

- background-color;
- background-image;
- background-repeat;
- background-attachment;
- background-position.

3.4.1 BACKGROUND COLOR

The background-color property specifies the background color of an element. The background color of a page is defined in the body selector:

EXAMPLE 72

```
body {
    background-color: #b0c4de;
}
```

With CSS, a color is most often specified by:

- a HEX value - like "#ff0000";
- an RGB value - like "rgb(255,0,0)";
- a color name - like "red".

In the example below, the h1, p, and div elements have different background colors:

EXAMPLE 73

```
h1 {
    background-color: #6495ed;
}

p {
    background-color: #e0ffff;
}

div {
```

```
background-color: #b0c4de;
}
```

3.4.2 BACKGROUND IMAGE

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element. The background image for a page can be set like this:

EXAMPLE 74

```
body {
    background-image: url("paper.gif");
}
```

Below is an example of a bad combination of text and background image. The text is almost not readable:

EXAMPLE 75

```
body {
    background-image: url("bgdesert.jpg");
}
```

By default, the background-image property repeats an image both horizontally and vertically. Some images should be repeated only horizontally or vertically, or they will look strange. If the image is repeated only horizontally (repeat-x), the background will look better:

EXAMPLE 76

```
body {
    background-image: url("gradient.png");
    background-repeat: repeat-x;
}
```

Showing the image only once is specified by the background-repeat property:

EXAMPLE 77

```
body {
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
}
```

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much. The position of the image is specified by the background-position property:

EXAMPLE 78

```
body {
```

```
background-image: url("img_tree.png");
background-repeat: no-repeat;
background-position: right top;
}
```

As you can see from the examples above, there are many properties to consider when dealing with backgrounds. To shorten the code, it is also possible to specify all the properties in one single property. This is called a shorthand property. The shorthand property for background is simply "background":

EXAMPLE 79

```
body {
    background: #ffffff url("img_tree.png") no-repeat right
top;
}
```

3.5 CSS TEXT

3.5.1 TEXT COLOR

The color property is used to set the color of the text. With CSS, a color is most often specified by:

- a HEX value - like "#ff0000";
- an RGB value - like "rgb(255,0,0)";
- a color name - like "red".

The default color for a page is defined in the body selector.

EXAMPLE 80

```
body {
    color: blue;
}

h1 {
    color: #00ff00;
}

h2 {
    color: rgb(255,0,0);
}
```

3.5.2 TEXT ALIGNMENT

The text-align property is used to set the horizontal alignment of a text. Text can be centered, or aligned to the left or right, or justified. When text-align is set to "justify", each

line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

EXAMPLE 81

```
h1 {
    text-align: center;
}

p.date {
    text-align: right;
}

p.main {
    text-align: justify;
}
```

3.5.3 TEXT DECORATION

The text-decoration property is used to set or remove decorations from text. The text-decoration property is mostly used to remove underlines from links for design purposes:

EXAMPLE 82

```
a {
    text-decoration: none;
}
```

It can also be used to decorate text:

EXAMPLE 83

```
h1 {
    text-decoration: overline;
}

h2 {
    text-decoration: line-through;
}

h3 {
    text-decoration: underline;
}
```

3.5.4 TEXT TRANSFORMATION

The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

EXAMPLE 84

```
p.uppercase {
    text-transform: uppercase;
}

p.lowercase {
    text-transform: lowercase;
}

p.capitalize {
    text-transform: capitalize;
}
```

3.5.5 TEXT INDENTATION

The text-indent property is used to specify the indentation of the first line of a text.

EXAMPLE 85

```
p {
    text-indent: 50px;
}
```

3.6 CSS FONT

3.6.1 FONT FAMILY

The font family of a text is set with the font-family property. The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font. Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available. If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman". More than one font family is specified in a comma-separated list:

EXAMPLE 86

```
p {
    font-family: "Times New Roman", Times, serif;
}
```

3.6.2 FONT STYLE

The font-style property is mostly used to specify italic text. This property has three values:

- normal - The text is shown normally;
- italic - The text is shown in italics;
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported).

EXAMPLE 87

```

p.normal {
    font-style: normal;
}

p.italic {
    font-style: italic;
}

p.oblique {
    font-style: oblique;
}

```

3.6.3 FONT SIZE

The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs. Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs. The font-size value can be an absolute, or relative size.

Absolute size:

- Sets the text to a specified size;
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons);
- Absolute size is useful when the physical size of the output is known.

Relative size:

- Sets the size relative to surrounding elements;
- Allows a user to change the text size in browsers

If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Setting the text size with pixels gives you full control over the text size:

EXAMPLE 88

```

h1 {
    font-size: 40px;
}

h2 {
    font-size: 30px;
}

p {
    font-size: 14px;
}

```

The example above allows Internet Explorer 9, Firefox, Chrome, Opera, and Safari to resize the text. The example above does not work in IE, prior version 9. The text can be

resized in all browsers using the zoom tool (however, this resizes the entire page, not just the text).

3.7 CSS LINKS

Links can be styled with any CSS property (e.g. color, font-family, background, etc.). In addition, links can be styled differently depending on what state they are in. The four links states are:

- a:link - a normal, unvisited link;
- a:visited - a link the user has visited;
- a:hover - a link when the user mouses over it;
- a:active - a link the moment it is clicked.

EXAMPLE 89

```
/* unvisited link */
a:link {
color: #FF0000;
}

/* visited link */
a:visited {
color: #00FF00;
}

/* mouse over link */
a:hover {
color: #FF00FF;
}

/* selected link */
a:active {
color: #0000FF;
}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited;
- a:active MUST come after a:hover.

3.8 CSS LISTS

In HTML, there are two types of lists:

- unordered lists - the list items are marked with bullets;
- ordered lists - the list items are marked with numbers or letters.

With CSS, lists can be styled further, and images can be used as the list item marker.

3.8.1 DIFFERENT LIST ITEM MARKERS

The type of list item marker is specified with the list-style-type property (some of the values are for unordered lists, and some for ordered lists):

EXAMPLE 90

```
ul.a {
    list-style-type: circle;
}

ul.b {
    list-style-type: square;
}

ol.c {
    list-style-type: upper-roman;
}

ol.d {
    list-style-type: lower-alpha;
}
```

3.8.2 AN IMAGE AS THE LIST ITEM MARKER

To specify an image as the list item marker, use the `list-style-image` property:

EXAMPLE 91

```
ul {
    list-style-image: url('sqpurple.gif');
}
```

The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari.

3.8.3 CROSSBROWSER SOLUTION

The following example displays the image-marker equally in all browsers:

EXAMPLE 92

```
ul {
    list-style-type: none;
    padding: 0px;
    margin: 0px;
}

ul li {
    background-image: url(sqpurple.gif);
    background-repeat: no-repeat;
    background-position: 0px 5px;
    padding-left: 14px;
}
```

Example explained:

- For ul:
 - Set the list-style-type to none to remove the list item marker;
 - Set both padding and margin to 0px (for cross-browser compatibility).
- For all li in ul:
 - Set the URL of the image, and show it only once (no-repeat);
 - Position the image where you want it (left 0px and down 5px);
 - Position the text in the list with padding-left.

3.8.4 LIST - SHORTHAND PROPERTY

It is also possible to specify all the list properties in one, single property. This is called a shorthand property. The shorthand property used for lists, is the list-style property:

EXAMPLE 93

```
ul {  
    list-style: square url("sqpurple.gif");  
}
```

When using the shorthand property, the order of the values are:

- list-style-type;
- list-style-position (for a description, see the CSS properties table below);
- list-style-image.

It does not matter if one of the values above are missing, as long as the rest are in the specified order.

3.9 CSS TABLES

To specify table borders in CSS, use the border property. The example below specifies a black border for table, th, and td elements:

EXAMPLE 94

```
table, th, td {  
    border: 1px solid black;  
}
```

Notice that the table in the example above has double borders. This is because both the table and the th/td elements have separate borders. To display a single border for the table, use the border-collapse property.

3.9.1 COLLAPSE BORDERS

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

EXAMPLE 95

```
table {  
    border-collapse: collapse;
```

```
}  
  
table, th, td {  
    border: 1px solid black;  
}
```

3.9.2 TABLE WIDTH AND HEIGHT

Width and height of a table is defined by the width and height properties. The example below sets the width of the table to 100%, and the height of the `th` elements to 50px:

EXAMPLE 96

```
table {  
    width: 100%;  
}  
  
th {  
    height: 50px;  
}
```

3.9.3 TABLE TEXT ALIGNMENT

The text in a table is aligned with the `text-align` and `vertical-align` properties. The `text-align` property sets the horizontal alignment, like left, right, or center:

EXAMPLE 97

```
td {  
    text-align: right;  
}
```

The `vertical-align` property sets the vertical alignment, like top, bottom, or middle:

EXAMPLE 98

```
td {  
    height: 50px;  
    vertical-align: bottom;  
}
```

3.9.4 TABLE PADDING

To control the space between the border and content in a table, use the `padding` property on `td` and `th` elements:

EXAMPLE 99

```
td {
```

```
padding: 15px;  
}
```

3.9.5 TABLE COLOR

The example below specifies the color of the borders, and the text and background color of th elements:

EXAMPLE 100




```
table, td, th {  
    border: 1px solid green;  
}  
  
th {  
    background-color: green;  
    color: white;  
}
```

4 PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP code can be simply mixed with HTML code, or it can be used in combination with various templating engines and web frameworks. PHP code is usually processed by a PHP interpreter, which is usually implemented as a web server's native module or a Common Gateway Interface (CGI) executable. After the PHP code is interpreted and executed, the web server sends resulting output to its client, usually in form of a part of the generated web page – for example, PHP code can generate a web page's HTML code, an image, or some other data. PHP has also evolved to include a command-line interface (CLI) capability and can be used in standalone graphical applications.

4.1 INSTALLATION OF A LOCAL SERVER

- Download EasyPHP from the website www.easyphp.org;
- double-click on the downloaded executable;
- select an installation directory and follow the instructions.

Before doing anything, verify that `easyphp.exe` is launched and servers are running. When EasyPHP is launched, an icon appears in the systray (beside the clock). If this icon looks like this  or like this  or like this , that means that servers are not running properly. If this is the case, double-click the icon to bring up the window below. Start the server(s) that didn't start (Figure 2) and wait until ...

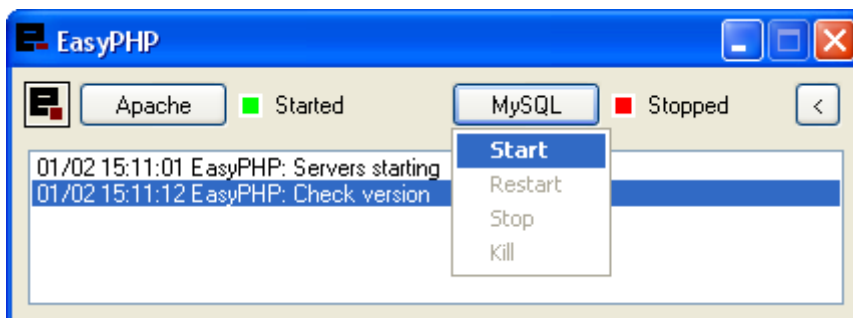


Figure 2: Start of MySQL.

... the window looks like (Figure 3):

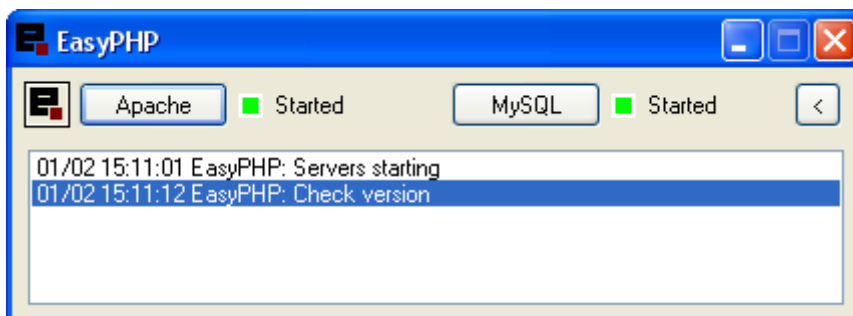




Figure 3: Apache and MySQL is running.

When everything is fine, the systray icon looks like : 

Now you can open the administration page. Right-click on  and select "Administration". The administration page allows you to manage PHP / Apache / MySQL, aliases, modules and to use the code tester.

Other menu options:

- Help: help on EasyPHP;
- Log Files: record errors generated by Apache, MySQL and EasyPHP;
- Configuration: gives access to various configuration tools;
- Explorer: open the directory "localweb" in Windows Explorer;
- Administration : opens the administration page;
- Web local : opens local web;
- Restart : restarts Apache and MySQL;
- Start/Stop : Starts/Stops Apache and MySQL;
- Quit : closes EasyPHP.

You must place your files either in the "localweb" directory, in an alias folder or a virtual host folder (module "Virtual Hosts Manager" needed). So that PHP can interpret your PHP pages.

4.2 YOUR FIRST PHP PAGE

There are many ways to program in PHP and there are many suitable text editors (eg. specialized for HTML or PHP with syntax highlighting etc). For this example, you can use a simple text editor.

- 1) Open a new file;
- 2) Type the structure (Example 101) of an HTML page

EXAMPLE 101

```
<html>
<head>
  <title>My first page in PHP.</title>
</head>
<body>

</body>
</html>
```

The Example 102 is designed to display the current date. The PHP Code is integrated directly into the HTML:

EXAMPLE 102

```
<html>
<head>
  <title>My first page in PHP.</title>
</head>
<body>

  Current date : <?php print(date("l F d, Y")); ?>
```

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

- 3) Create a new directory in the "localweb" directory (or in an alias folder, or in a virtual host folder). Save your first PHP page there with the following extension: .php. Name it "date.php". Make sure, in the Windows Folder Options, that the extensions of the files whose type is known are visible.
- 4) If you want to see the result you need to use a browser. From the administration page select your folder (localweb, alias or virtual host) and click on your file "date.php". Your browser will display a page with the current date; for example: "current Date: Wednesday March 22, 2013".

4.3 PHP BASICS

PHP is an easy-to-understand language and it is enough to be familiar with the basics of the language in many situations. With PHP, data can be stored, updated or deleted. Everything happens on a web server, where all code sources are stored. Firstly, a PHP script is processed on web server and its result is sent to the client browser. It means that, for example, 300/30 is computed and then only the resulting number 10 is sent to a browser. This is why you find in the source code of a web page only "10" (this is the difference between PHP and JavaScript, where code is processed directly in the browser). Thus, a PHP source code cannot be displayed in the browser.

With PHP, it is possible to create a website discussion forum, a guestbook, a counter, a poll or a chart. You also have an option to link your site with a database such as MySQL.

There exists at least one PHP function that is common perhaps for each site. Each web page on the site usually consists from several parts, like the header, links, menu or the footer, which are the same on most of the pages. In this case, it is better to implement those parts in separate files, which are later included to the webpages using special PHP statement (see Section PHP Include Files for details). In this manner, when something changes in, for example, left menu, only the left menu file is updated, and the change is propagated to all the webpages that includes the left menu.

Any PHP script must be enclosed in special tags used for this purpose. The first option is to enclose all the script between `<? and ?>`. The following script displays the text in quotation marks.

EXAMPLE 103

```
<? echo "My first PHP script."; ?>
```

The second option is a little bit longer. If one wants to use PHP to generate XML-documents, use the opening tag `<?php`. An example of the second possibility:

EXAMPLE 104

```
<?php echo "My second PHP script."; ?>
```

The most important expression in PHP is semicolon (;). Each function, line, declaration must be separated by the semicolon. It can be compared to the CSS (when properties omit the semicolon, the browser does not understand them). Whenever a script is not working, try to check, where semicolon has been forgotten.

It is useful to write comments in source code. Comments are the text that is omitted by a browser. Example:

EXAMPLE 105

```
<?php
/*
multi-line
comment
*/

// one line comment
?>
```

The most common command used for output is `echo()`, or, its less common alternative `print()`:

EXAMPLE 106

```
<?php
echo ("this is text" .
"on two lines<br>");
echo ("display text <br />");
echo "without the brackets<br>";
print ("the same</br>");
echo ("<strong>you can also use HTML tags</strong></br>");
echo 'you can use single quotation marks';
?>
```

Multiline text should be split and merged by the dot character. The previous script in the browser outputs:

```
this is text on two lines
display text
without the brackets
the same
you can also use HTML tags
you can use single quotation marks
```

Whenever we output HTML tags with special characters like, e.g., `<`, `>`, etc., we must substitute them by their convenient alternatives. For example, when we output `<body bgcolor="red">`, we must put in PHP echo:

EXAMPLE 107

```
echo ("&lt;bodybgcolor=\"red\"&gt;");
```

Other example:

EXAMPLE 108

```
echo "This \" is quotation mark";
```

4.4 VARIABLES

It is hard to imagine a sophisticated PHP script without variables. In PHP, we declare the variables starting with the dollar sign \$. Example:

EXAMPLE 109

```
$dollar = "1$";
$day="wednesday";
```

The name of a variable is given right after the \$ (without any spaces). We have introduced two variables: the dollar and the day. Their values are written after the equal sign. The name of a variable in PHP is case sensitive. Thus, this does not work:

```
$vari = "abc";
echo $Vari;
```

Variables are displayed using echo () :

EXAMPLE 110

```
$dollar="1$";
echo($dollar);
echo("$dollar");
echo("I have just " . $dolar . "<br />");
```

Variables can be changed using PHP operators.

Table 1: The operators in PHP

Arithmetic operators	
\$a+\$b	adds \$a to \$b
\$a-\$b	subtracts \$b from \$a
\$a/\$b	divide \$a by \$b
\$a*\$b	multiplies \$a by \$b
Assignment operators	
\$a++	\$a = \$a + 1
\$a--	\$a = \$a - 1
\$a += \$b	\$a = \$a + \$b
\$a -= \$b	\$a = \$a - \$b
\$a *= \$b	\$a = \$a * \$b
\$a /= \$b	\$a = \$a / \$b
Comparison operators	
\$a == \$b	\$a is equal to \$b
\$a != \$b	\$a is not equal to \$b
\$a > \$b	\$a is higher than \$b
\$a < \$b	\$a is lower than \$b
Logical operators	
\$a \$b	logical, or; returns 1 if at least one of the variables is true
\$a && \$b	logical and; returns 1 if both of the variables are true

!\$a	negation of \$a
------	-----------------

In logical comparison, `true` has the value 1 and `false` has the value 0.
Variables in PHP are of different types:

Table 2: Variables types

Type	Meaning
String	E.g. <code>\$mystring= "plain text";</code>
Integer	Integer, e.g., <code>\$myint = 2;</code>
Float, real or double	Float number, e.g. 3.14, 0.9, 2.78
Boolean	Logical variable, values TRUE, FALSE (1, 0), written as TRUE or FALSE

Different types of variables are differently introduced:

EXAMPLE 111

```
$mystring = "plain text";
$myint = 2;
$myfloat = 0.2;
$mylogical = TRUE;
```

4.5 CONDITIONAL STATEMENTS

Conditional statements manage the flow of the script. Those statements are `if`, `elseif` and `switch`.

4.5.1 IF

Statement `if` checks if the given condition is true.

EXAMPLE 112

```
$a=1;
$b=2;
if($a == $b){
echo ("Variables a and b are equal.");
}
```

If the variables `$a` and `$b` were be equal, the script would display: `Variables a and b are equal.` However, it would not happen as the given condition (`$a==$b`) is not valid (true).

4.5.2 IF...ELSE

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

EXAMPLE 113

```

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

```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

EXAMPLE 114

```

<?php
$t=date("H");
if ($t<"20")
{
echo "Have a good day!";
}
else
{
echo "Have a good night!";
}
?>

```

4.5.3 IF...ELSEIF....ELSE

Use the `if...else if...else` statement to select one of several blocks of code to be executed. Syntax:

EXAMPLE 115

```

if (condition1)
{
code to be executed if condition1 is true;
}
else if (condition2)
{
code to be executed if condition1 is false and condition2
is true;
}
else
{
code to be executed if the both conditions are false;
}

```

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

EXAMPLE 116

```
<?php
$t=date("H");
if ($t<"10")
{
echo "Have a good morning!";
}
else if ($t<"20")
{
echo "Have a good day!";
}
else
{
echo "Have a good night!";
}
?>
```

4.5.4 SWITCH STATEMENT

Use the switch statement to select one of many blocks of code to be executed. Syntax:

EXAMPLE 117

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

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. The default statement is used if no match is found.

EXAMPLE 118

```
<?php
$favcolor="red";
switch ($favcolor)
{
case "red":
echo "Your favorite color is red!";
```

```

break;
case "blue":
echo "Your favorite color is blue!";
break;
case "green":
echo "Your favorite color is green!";
break;
default:
echo "Your favorite color is neither red, blue, or green!";
}
?>

```

4.6 LOOPS

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 PHP, we have the following looping statements:

- `while` - loops through a block of code while a specified condition is true
- `do...while` - loops through a block of code once, and then repeats the loop as long as a specified 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

4.6.1 THE WHILE LOOP

The while loop executes a block of code while a condition is true. Syntax:

EXAMPLE 119

```

while (condition)
{
code to be executed;
}

```

The example below first sets a variable `i` to 1 (`$i=1;`). Then, the while loop will continue to run as long as `i` is less than, or equal to 5. `i` will increase by 1 each time the loop runs:

EXAMPLE 120

```

<html>
<body>

<?php
$i=1;
while($i<=5)

```

```
{
echo "The number is " . $i . "<br>";
  $i++;
}
?>

</body>
</html>
```

Output is:

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

4.6.2 THE DO...WHILE STATEMENT

The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true. Syntax:

EXAMPLE 121

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

The example below first sets a variable *i* to 1 ($\$i=1;$).

Then, it starts the do...while loop. The loop will increment the variable *i* with 1, and then write some output. Then the condition is checked (is *i* less than, or equal to 5), and the loop will continue to run as long as *i* is less than, or equal to 5:

EXAMPLE 122

```
<html>
<body>

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

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

Output is:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

4.6.3 THE FOR LOOP

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

Syntax:

EXAMPLE 123

```
for (init; condition; increment)
{
code to be executed;
}
```

Parameters:

init: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)

condition: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.

increment: Mostly used to increment a counter (but can be any code to be executed at the end of the iteration)

Note: The init and increment parameters above can be empty or have multiple expressions (separated by commas).

The example below defines a loop that starts with `i=1`. The loop will continue to run as long as the variable `i` is less than, or equal to 5. The variable `i` will increase by 1 each time the loop runs:

EXAMPLE 124

```
<html>
<body>

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



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

Output:

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

4.6.4 THE FOREACH LOOP

The foreach loop is used to loop through arrays. Syntax:

EXAMPLE 125

```
foreach ($array as $value)  
{  
  code to be executed;  
}
```

For every loop iteration, the value of the current array element is assigned to \$value (and the array pointer is moved by one) - so on the next loop iteration, you'll be looking at the next array value.

The following example demonstrates a loop that will print the values of the given array:

EXAMPLE 126

```
<html>  
<body>  
  
<?php  
$x=array("one","two","three");  
foreach ($x as $value)  
{  
  echo $value . "<br>";  
}  
?>  
  
</body>  
</html>
```

Output is:

```
one  
two  
three
```

4.7 PHP FUNCTIONS

The real power of PHP comes from its functions. In PHP, there are more than 700 built-in functions.

4.7.1 PHP FUNCTIONS

In this chapter we will show you how to create your own functions. To keep the script from being executed when the page loads, you can put it into a function. A function will be executed by a call to the function. You may call a function from anywhere within a page.

4.7.2 CREATE A PHP FUNCTION

A function will be executed by a call to the function. Syntax:

EXAMPLE 127

```
function functionName()
{
code to be executed;
}
```

PHP function guidelines:

- Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)

A simple function that writes my name when it is called:

EXAMPLE 128

```
<html>
<body>

<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes
```

4.7.3 PHP FUNCTIONS – ADDING PARAMETERS

To add more functionality to a function, we can add parameters. A parameter is just like a variable. Parameters are specified after the function name, inside the parentheses.

The following example will write different first names, but equal last name:

EXAMPLE 129

```
<html>
<body>

<?php
Function writeName($fname)
{
echo $fname . " Refsnes.<br>";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes.
My sister's name is HegeRefsnes.
My brother's name is Stale Refsnes.
```

The following function has two parameters:

EXAMPLE 130

```
<html>
<body>

<?php
Function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br>";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
```

```

writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>

```

Output:

```

My name is Kai Jim Refsnes.
My sister's name is HegeRefsnes!
My brother's name is StåleRefsnes?

```

4.7.4 PHP FUNCTIONS - RETURN VALUES

To let a function return a value, use the return statement.

EXAMPLE 131

```

<html>
<body>

<?php
function add($x,$y)
{
$total=$x+$y;
return $total;
}

echo "1 + 16 = " . add(1,16);
?>

</body>
</html>

```

Output:

```
1 + 16 = 17
```

4.8 ARRAYS

An array stores multiple values in one single variable. Example:

EXAMPLE 132

```

<?php
$cars=array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " .

```

```
$cars[2] . " .";
?>
```

What is an Array? An array is a special variable, which can hold more than one value at a time. If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Volvo";
$cars2="BMW";
$cars3="Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300? The solution is to create an array! An array can hold many values under a single name, and you can access the values by referring to an index number.

In PHP, the `array()` function is used to create an array:

EXAMPLE 133

```
$myarray = array();
```

In PHP, there are three types of arrays:

- Indexed arrays - Arrays with numeric index
- Associative arrays - Arrays with named keys
- Multidimensional arrays - Arrays containing one or more arrays

4.8.1 INDEXED ARRAYS

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0):

EXAMPLE 134

```
$cars=array("Volvo", "BMW", "Toyota");
```

or the index can be assigned manually:

EXAMPLE 135

```
$cars[0]="Volvo";
$cars[1]="BMW";
$cars[2]="Toyota";
```

The following example creates an indexed array named `$cars`, assigns three elements to it, and then prints a text containing the array values:

EXAMPLE 136

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " .
$cars[2] . ".";
?>
```

4.8.2 GET THE LENGTH OF AN ARRAY - THE COUNT() FUNCTION

The count () function is used to return the length (the number of elements) of an array:

EXAMPLE 137

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo count($cars);
?>
```

4.8.3 LOOP THROUGH AN INDEXED ARRAY

To loop through and print all the values of an indexed array, you could use a for loop, like this:

EXAMPLE 138

```
<?php
$cars=array("Volvo","BMW","Toyota");
$arrlength=count($cars);

for($x=0;$x<$arrlength;$x++)
{
    echo $cars[$x];
    echo "<br>";
}
?>
```

4.8.4 ASSOCIATIVE ARRAYS

Associative arrays are arrays that use named keys that you assign to them. There are two ways to create an associative array:

EXAMPLE 139

```
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
```

or:

EXAMPLE 140

```
$age['Peter']="35";
$age['Ben']="37";
$age['Joe']="43";
```

The named keys can then be used in a script:

EXAMPLE 141

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

4.8.5 LOOP THROUGH AN ASSOCIATIVE ARRAY

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

EXAMPLE 142

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

foreach($age as $x=>$x_value)
{
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

4.9 PHP INCLUDE FILES

In PHP, you can insert the content of one PHP file into another PHP file before the server executes it. Include and require statements are used to insert useful codes written in other files, in the flow of execution.

Include and require are identical, except upon failure:

- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use include. Otherwise, in case of Framework, CMS or a complex PHP application coding, always use require to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file. Syntax:

EXAMPLE 143

```
include 'filename';

or

require 'filename';
```

4.9.1 PHP INCLUDE AND REQUIRE STATEMENT

Let see a basic example. Assume that you have a standard header file, called "header.php". To include the header file in a page, use include/require:

EXAMPLE 144

```
<html>
<body>

<?php include 'header.php'; ?>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>

</body>
</html>
```

In another example, assume we have a standard menu file that should be used on all pages, e.g., "menu.php".

EXAMPLE 145

```
echo '<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>';
```

All pages in the Web site should include this menu file. Here is how it can be done:

EXAMPLE 146

```
<html>
<body>

<div class="leftmenu">
<?php include 'menu.php'; ?>
</div>
```



```
<h1>Welcome to my home page.</h1>
<p>Some text.</p>

</body>
</html>
```

In the third example, assume we have an include file with some variables defined ("vars.php"):

EXAMPLE 147

```
<?php
$color='red';
$car='BMW';
?>
```

Then the variables can be used in the calling file:

EXAMPLE 148

```
<html>
<body>

<h1>Welcome to my home page.</h1>
<?php include 'vars.php';
echo "I have a $color $car"; // I have a red BMW
?>

</body>
</html>
```

5 FORM HANDLING USING PHP

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts. The example below contains an HTML form with two input fields and a submit button:

EXAMPLE 149

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="fname">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>
```

When a user fills out the form above and clicks on the submit button, the form data is sent to a PHP file, called "welcome.php":

"welcome.php" looks like this:

EXAMPLE 150

```
<html>
<body>

Welcome <?php echo $_POST["fname"]; ?>!<br>
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

Output could be something like this:

```
Welcome John!
You are 28 years old.
```

5.1 PHP \$_GET VARIABLE

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

EXAMPLE 151

```
<form action="welcome.php" method="get">
```

```
Name: <input type="text" name="fname">  
Age: <input type="text" name="age">  
<input type="submit">  
</form>
```

When the user clicks the "Submit" button, the URL sent to the server could look something like this:

EXAMPLE 152

```
http://www.domain.com/welcome.php?fname=Peter&age=37
```

The "welcome.php" file can now use the `$_GET` variable to collect form data (the names of the form fields will automatically be the keys in the `$_GET` array):

EXAMPLE 153

```
Welcome <?php echo $_GET["fname"]; ?>.<br>  
You are <?php echo $_GET["age"]; ?> years old!
```

When using `method="get"` in HTML forms, all variable names and values are displayed in the URL. Thus, this method should not be used when sending passwords or other sensitive information! However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

Note: The get method is not suitable for very large variable values. It should not be used with values exceeding 2000 characters.

5.2 PHP `$_POST` FUNCTION

The predefined `$_POST` variable is used to collect values from a form sent with `method="post"`. Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send. However, there is an 8 MB max size for the POST method, by default (can be changed by setting the `post_max_size` in the `php.ini` file).

EXAMPLE 154

```
<form action="welcome.php" method="post">  
Name: <input type="text" name="fname">  
Age: <input type="text" name="age">  
<input type="submit">  
</form>
```

When the user clicks the "Submit" button, the URL will look like this:

EXAMPLE 155

```
http://www.domain.com/welcome.php
```

The "welcome.php" file can now use the \$_POST variable to collect form data (the names of the form fields will automatically be the keys in the \$_POST array):

EXAMPLE 156

```
Welcome <?php echo $_POST["fname"]; ?>!<br>
You are <?php echo $_POST["age"]; ?> years old.
```

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send. However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

5.3 PHP \$_REQUEST FUNCTION

The predefined \$_REQUEST variable contains the contents of both \$_GET, \$_POST, and \$_COOKIE. The \$_REQUEST variable can be used to collect form data sent with both the GET and POST methods.

EXAMPLE 157

```
Welcome <?php echo $_REQUEST["fname"]; ?>!<br>
You are <?php echo $_REQUEST["age"]; ?> years old.
```

6 MYSQL

6.1 CREATE DATABASE

A database is a way to store lots of information. You might want to store the names and addresses of all your contacts, or save usernames and passwords for your online forum or maybe customer information.

In a database, you save the information in a Table. A single database can contain many tables, and they can be linked together. When the tables are linked together, it's said to be a relational database. If you just have a single table in your database, then it's called a flat-file database. Flat-file database are easier to create and understand, so we'll start by creating one of these using phpMyAdmin.

So bring up phpMyAdmin, if you haven't already done so.

Although it looks a bit muddled, the part to concentrate on is the textbox under the words create new database, as in the next image (Figure 4):

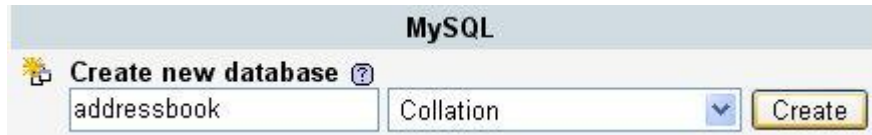


Figure 4: Editing a name of the new database.

After you have typed a name for your new database, click the "Create" button. You will be taken to a new area (Figure 5):

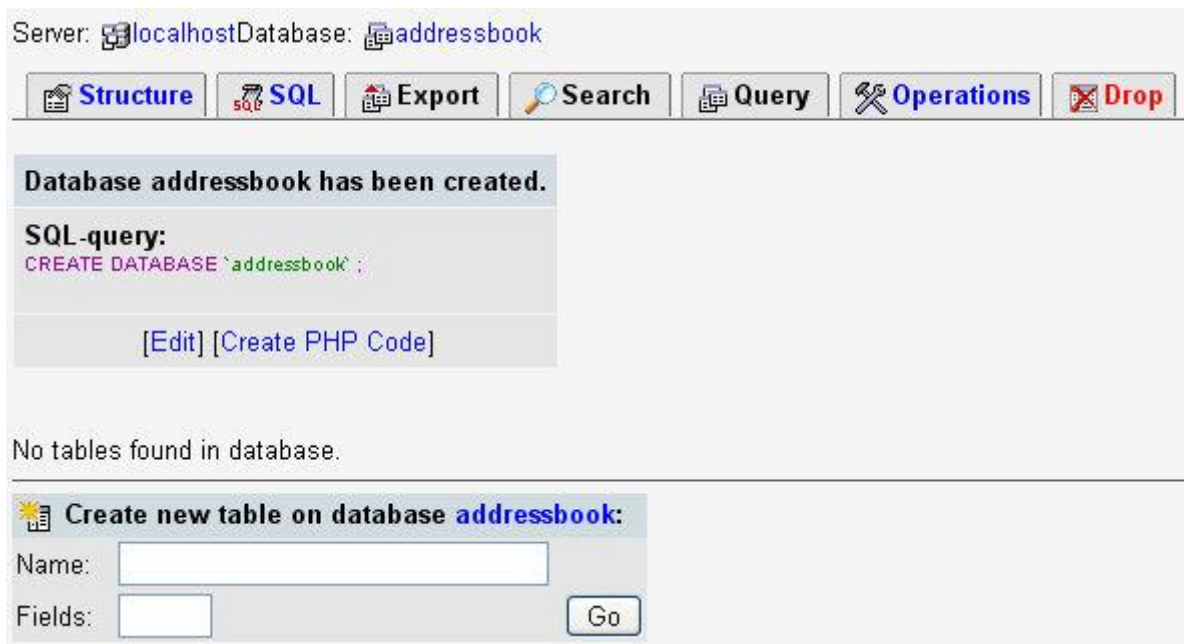


Figure 5: Editing name and fields of a table.

In this new area, you can create a Table to go in your database. At the moment, as it says, there are **No tables found in the database**. But the database itself has been created.

To create a new table, type a name for it in the box at the bottom. You can also type a number for the **Fields** textbox. You can insert for example (Figure 6):

Create new table on database **addressbook**:

Name:

Fields:

Figure 6: Example of filling the name and number of fields.

When you've finished, click the **Go** button. Another, more complex, area will appear (Figure 7):

Server: localhost Database: @addressbook Table: tbl_address_book

Field	Type	Length/Values	Collation	Attributes	Null	Default	Extra
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		

Table comments:

Table type:

Collation:

Add field(s)

* If field type is "enum" or "set", please enter the values using this format: 'a','b','c'...

If you ever need to put a backslash ("\") or a single quote (") amongst those values, backslash it (for example '\xyz' or 'a\b').

** For default values, please enter just a single value, without backslash escaping or quotes, using this format: a

Figure 7: Editing the fields in database.

In this new area, you set up the **fields** in your database. You can specify whether a field is for text, for numbers, for yes/no values, etc.

Although they are set out in rows in the images, the rows are actually the Columns you saw earlier – the Fields. Each Field needs a name. So go ahead and type the following for your Field names:

Field	Type
<input type="text" value="ID"/>	VARCHAR
<input type="text" value="First_Name"/>	VARCHAR
<input type="text" value="Surname"/>	VARCHAR
<input type="text" value="Address"/>	VARCHAR

Figure 8: Editing of the Fields names and Types.

So we have given each column in our table a name: **ID**, **First_Name**, **Surname**, and **Address**. The next thing to set is what type of data will be going in to each field - do you want to store text in this field, numbers, Yes/No value, etc?

To set the type of data going into a field, you select an item from the **Type** drop down list. Click the down arrow to see the following list you can choose from (Figure 9):



Figure 9: A List of the Field Types.

As you can see from the image above, there's quite a lot! But you won't use most of them. For the values we have in our four fields, we want to hold these Types:

- **ID** – A number, used just to identify each record. This needs to be unique for each record;
- **First_Name** – Text;
- **Surname** – Text;
- **Address** – Text.

If you look at the list, there is an INT but no Number; and there are four different Text Types to choose from. We can use INT (meaning integer) for the numbers, but again, there are a few Integer Types to choose from. And that's leaving out things like float and double. Here's the difference between them, though.

6.1.1 INTEGER VALUES

- **TINYINT** Signed: -128 to 127. Unsigned: 0 to 255;
- **SMALLINT** Signed: -32768 to 32767. Unsigned: 0 to 65535;
- **MEDIUMINT** Signed: -8388608 to 8388607. Unsigned: 0 to 16777215;
- **INT** Signed: -2147483648 to 2147483647. Unsigned: 0 to 4294967295;
- **BIGINT** Signed: -9223372036854775808. Unsigned: 0 to 18446744073709551615.

The signed and unsigned are for minus and non minus values. So if you need to store negative values, you need to be aware of the signed ranges. If you were using a TINYINT value, for example, you can go from minus 128 to positive 127. If you didn't need the minus value, you can go from 0 to positive 255.

For our address book, we have an ID field. We're using this just to identify a record (row). Each record will be unique, so it will need a different number for each. We can set it to one of the INT values. But which one?

If we set ID to TINYINT, then you'd run in to problem if you tried to store more than 255 records. If you used SMALLINT, you'd have problems if you tried to stored the details of

friend number 65536. IF you have more than 65 and half thousand friends, then you need a different INT type. We'll assume that you don't, so we'll use SMALLINT.

We've only set Lengths for the VARCHAR TYPES. If you leave it blank for VARCHAR, you'll get a default value of 1 character.

6.1.2 TEXT TYPES

The length for the text types can be quite confusing. The MySQL manual says this about the various lengths that each text type can hold:

- **TINYTEXT** L+1 byte, where $L < 2^8$;
- **TEXT** L+2 bytes, where $L < 2^{16}$;
- **MEDIUMTEXT** L+3 bytes, where $L < 2^{24}$;
- **LONGTEXT** L+4 bytes, where $L < 2^{32}$.

This is not terribly helpful for beginners! So what does it mean. Well, the L + 1 part means, "The length of the string, plus 1 byte to store the value." The translated values for each are approximately:

- **TINYTEXT** 256 bytes;
- **TEXT** 64 KiloBytes;
- **MEDIUMTEXT** 16 MegaBytes;
- **LONGTEXT** 4 GigaBytes.

To confuse the issue even more, you can also use CHAR and VARCHAR to store your text. These are quite useful, if you know how many characters you want to store. For example, for a UK postcode you don't need more than 9 characters, and one of those will be a blank space. So there's no sense in setting a postcode field to hold 4 gigabytes! Instead, use CHAR or VARCHAR.

6.1.3 CHAR

You specify how many characters you want the field to hold. The maximum value is 255. For example:

CHAR(10)

This field can then hold a maximum of ten characters. But if you only use 4 of them, the rest of the 10 characters will be blank spaces. The blank spaces get added to the right of your text:

"TEXT "

"TENLETTERS"

VARCHAR

Like CHAR, but the rest of the characters are not padded with blank spaces. The maximum value before MySQL 5.0.3 was 255. After this it's jumped to 65, 535. With VARCHAR, there is also an extra byte that records how long your text is.

For our fields, then, we'll use the following Types:

- **ID** SMALLINT;
- **First_Name** VARCHAR;
- **Surname** VARCHAR;
- **Address** TINYTEXT.

So select these from your Types drop down list:

Field	Type ?	Length/Values*
ID	SMALLINT	
First_Name	VARCHAR	50
Surname	VARCHAR	50
Address	TINYTEXT	

Figure 10: Editing of the Tables and Length / Values.

We've only set Lengths for the VARCHAR TYPES. If you leave it blank for VARCHAR, you'll get a default value of 1 character.

The other Fields are **Null**, **Default** and **Extra**.

- **Null** - This is an important field in database terminology. It essentially means, "Should the field contain anything?" If you set a field to NOT NULL, then you can't leave it blank when you come to adding records to your database. Otherwise you'll get errors.
- **Default** - Do you want to add anything to the field, just in case it's left blank when adding a record? If so, type it in here.
- **Extra** - This is where you can set an auto increment value. This means adding one to the previous record number. This is ideal for us, as we have an ID field. Then we don't have to worry about this field. MySQL will take care of updating it for us.



Figure 11: Icons Primary Key, Index, and Unique.

The three icons are Primary Key, Index, and Unique (Figure 11). Primary keys are not terribly important for flat-file databases like ours. But they are important when you have more than one table, and want to link information. They are set to unique values, like our ID field. An index is useful for sorting information in your tables, as they speed things up. Unique is useful for those fields when there can't be any duplicate values.

So, set a primary key for the ID field by selecting the radio button, and choose Auto Increment. Your field screen then, minus the parts we've ignored, should look like this:

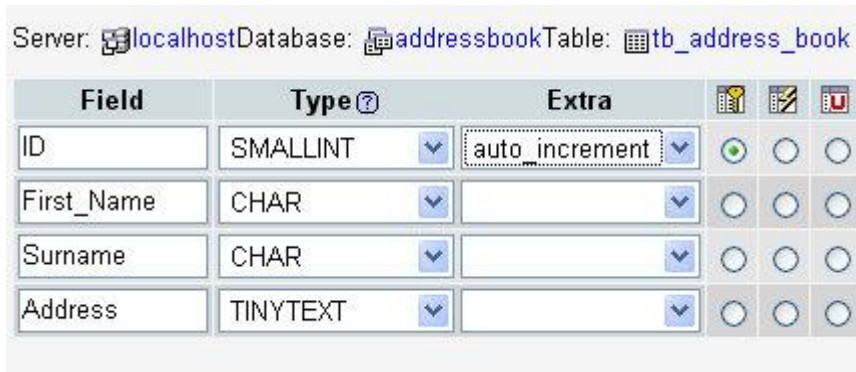


Figure 12: Editing of the Fields, Types and Extra.

Bear in mind what we've done here: we've just set up the fields for our table, and specified the kind of information that will be going into each field (the columns). We haven't yet added any information to the table.

Click the Save button on the fields screen. You'll be taken back to the Structure screen. There should be a lot more information there now. Don't worry if it looks a bit confusing. All we want to do is to add one record to the table. We'll then use PHP code to add some more.

6.2 CONNECT TO THE MySQL SERVER

6.2.1 OPEN A CONNECTION TO THE MYSQL SERVER

Before we can access data in a database, we must open a connection to the MySQL server. In PHP, this is done with the `mysqli_connect()` function. Syntax:

EXAMPLE 158

```
mysqli_connect(host,username,password,dbname);
```

Table 2: The connection parameters

Parameter	Description
host	Optional. Either a host name or an IP address
username	Optional. The MySQL user name
password	Optional. The password to log in with
dbname	Optional. The default database to be used when performing queries

Note: There are more available parameters, but the ones listed above are the most important.

In the following example we store the connection in a variable (`$con`) for later use in the script:

EXAMPLE 159

```
<?php
// Create connection
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
```

```

// Check connection
if (mysqli_connect_errno($con))
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}
?>

```

6.2.2 CLOSE A CONNECTION

The connection will be closed automatically when the script ends. To close the connection before, use the `mysqli_close()` function:

EXAMPLE 160

```

<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;

// Check connection
if (mysqli_connect_errno($con))
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

mysqli_close($con);
?>

```

6.3 CREATE DATABASE AND TABLES

6.3.1 CREATE A DATABASE

The `CREATE DATABASE` statement is used to create a database table in MySQL. We must add the `CREATE DATABASE` statement to the `mysqli_query()` function to execute the command. The following example creates a database named "my_db":

EXAMPLE 161

```

<?php
$con=mysqli_connect("example.com","peter","abc123");
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

```

```
// Create database
$sql="CREATE DATABASE my_db";
if (mysqli_query($con,$sql))
{
echo "Database my_db created successfully";
}
else
{
echo "Error creating database: " . mysqli_error($con);
}
?>
```

6.3.2 CREATE A TABLE

The CREATE TABLE statement is used to create a table in MySQL. We must add the CREATE TABLE statement to the `mysqli_query()` function to execute the command. The following example creates a table named "Persons", with three columns. The column names will be "FirstName", "LastName" and "Age":

EXAMPLE 162

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

// Create table
$sql="CREATE TABLE Persons(FirstName CHAR(30),LastName
CHAR(30),Age INT) ";

// Execute query
if (mysqli_query($con,$sql))
{
echo "Table persons created successfully";
}
else
{
echo "Error creating table: " . mysqli_error($con);
}
?>
```

Note: When you create a database field of type CHAR, you must specify the maximum length of the field, e.g. CHAR(50).

The data type specifies what type of data the field can hold. We describe only 2 data types, which are needed for our purposes in this text.

Table 3: Data types

Data type	Description
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED (The integer types have an extra option called UNSIGNED. Normally, the integer goes from an negative to positive value. Adding the UNSIGNED attribute will move that range up so it starts at zero instead of a negative number). The maximum number of digits may be specified in parenthesis
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. Note: If you put a greater value than 255 it will be converted to a TEXT type

6.3.3 PRIMARY KEYS AND AUTO INCREMENT FIELDS

Each table in a database should have a primary key field. A primary key is used to uniquely identify the rows in a table. Each primary key value must be unique within the table. Furthermore, the primary key field cannot be null because the database engine requires a value to locate the record.

The following example sets the PID field as the primary key field. The primary key field is often an ID number, and is often used with the AUTO_INCREMENT setting. AUTO_INCREMENT automatically increases the value of the field by 1 each time a new record is added. To ensure that the primary key field cannot be null, we must add the NOT NULL setting to the field:

EXAMPLE 163

```
$sql = "CREATE TABLE Persons
(
PID INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY(PID) ,
FirstNameCHAR(15) ,
LastNameCHAR(15) ,
Age INT
)";
```

6.4 INSERT INTO

The SQL statement INSERT INTO is used to insert new records in a table.

6.4.1 INSERT DATA INTO A DATABASE TABLE

The INSERT INTO statement is used to add new records to a database table. It is possible to write the INSERT INTO statement in two forms. The first form doesn't specify the column names where the data will be inserted, only their values:

EXAMPLE 164

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

The second form specifies both the column names and the values to be inserted:

EXAMPLE 165

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

To get PHP to execute the statements above we must use the `mysqli_query()` function. This function is used to send a query or command to a MySQL connection.

In the previous chapter we created a table named "Persons", with three columns; "FirstName", "LastName" and "Age". We will use the same table in this example. The following example adds two new records to the "Persons" table:

EXAMPLE 166

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

mysqli_query($con,"INSERT INTO Persons (FirstName,
LastName, Age)
VALUES ('Peter', 'Griffin',35)");

mysqli_query($con,"INSERT INTO Persons (FirstName,
LastName, Age)
VALUES ('Glenn', 'Quagmire',33)");

mysqli_close($con);
?>
```

6.4.2 INSERT DATA FROM A FORM INTO A DATABASE

Now we will create an HTML form that can be used to add new records to the "Persons" table. Here is the HTML form:

EXAMPLE 167

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

```

<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname">
Lastname: <input type="text" name="lastname">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>

```

When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php". The "insert.php" file connects to a database, and retrieves the values from the form with the PHP \$_POST variables. Then, the mysqli_query() function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.

Here is the "insert.php" page:

EXAMPLE 168

```

<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

$sql="INSERT INTO Persons (FirstName, LastName, Age)
VALUES
('$ _POST[firstname]','$ _POST[lastname]','$ _POST[age]')";

if (!mysqli_query($con,$sql))
{
die('Error: ' . mysqli_error($con));
}
echo "1 record added";

mysqli_close($con);
?>

```

6.5 SELECT

6.5.1 SELECT DATA FROM A DATABASE TABLE

The SELECT statement is used to select data from a database. Syntax:

EXAMPLE 169

```
SELECT column_name(s)
FROM table_name
```

To get PHP to execute the statement above we must use the `mysqli_query()` function. This function is used to send a query or command to a MySQL connection.

The following example selects all the data stored in the "Persons" table (The * character selects all the data in the table):

EXAMPLE 170

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM Persons");

while($row = mysqli_fetch_array($result))
{
echo $row['FirstName'] . " " . $row['LastName'];
echo "<br>";
}

mysqli_close($con);
?>
```

The example above stores the data returned by the `mysqli_query()` function in the `$result` variable. Next, we use the `mysqli_fetch_array()` function to return the first row from the recordset as an array. Each call to `mysqli_fetch_array()` returns the next row in the recordset. The while loop loops through all the records in the recordset. To print the value of each row, we use the PHP `$row` variable (`$row['FirstName']` and `$row['LastName']`). The output of the code above will be:

```
Peter Griffin
Glenn Quagmire
```

6.5.2 DISPLAY THE RESULT IN AN HTML TABLE

The following example selects the same data as the example above, but will display the data in an HTML table:

EXAMPLE 171

```

<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM Persons");

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";

while($row = mysqli_fetch_array($result))
{
echo "<tr>";
echo "<td>" . $row['FirstName'] . "</td>";
echo "<td>" . $row['LastName'] . "</td>";
echo "</tr>";
}
echo "</table>";

mysqli_close($con);
?>

```

The output of the code above will be:

Table 4: The output table

Firstname	Lastname
Glenn	Quagmire
Peter	Griffin

6.6 THE WHERE CLAUSE

The WHERE clause is used to filter records. Syntax:

EXAMPLE 172

```

SELECT column_name(s)
FROM table_name
WHERE column_name operator value

```

The following example selects all rows from the "Persons" table where "FirstName='Peter'":

EXAMPLE 173

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM Persons
WHERE FirstName='Peter'");

while($row = mysqli_fetch_array($result))
{
echo $row['FirstName'] . " " . $row['LastName'];
echo "<br>";
}
?>
```

The output of the code above will be:

```
Peter Griffin
```

6.7 ORDER BY KEYWORD

The ORDER BY keyword is used to sort the data in a recordset. The ORDER BY keyword sort the records in ascending order by default. If you want to sort the records in a descending order, you can use the DESC keyword. Syntax:

EXAMPLE 174

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

The following example selects all the data stored in the "Persons" table, and sorts the result by the "Age" column:

EXAMPLE 175

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
// Check connection
```

```
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM Persons ORDER BY
age");

while($row = mysqli_fetch_array($result))
{
echo $row['FirstName'];
echo " " . $row['LastName'];
echo " " . $row['Age'];
echo "<br>";
}

mysqli_close($con);
?>
```

The output of the code above will be:

```
Glenn Quagmire 33
Peter Griffin 35
```

It is also possible to order by more than one column. When ordering by more than one column, the second column is only used if the values in the first column are equal:

EXAMPLE 176

```
SELECT column_name(s)
FROM table_name
ORDER BY column1, column2
```

6.8 THE UPDATE STATEMENT

The UPDATE statement is used to update existing records in a table.Syntax:

EXAMPLE 177

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Earlier in the text we created a table named "Persons". Here is how it looks:

Table 5: The table Persons

FirstName	LastName	Age
Peter	Griffin	35
Glenn	Quagmire	33

The following example updates some data in the "Persons" table:

EXAMPLE 178

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

mysqli_query($con,"UPDATE Persons SET Age=36
WHERE FirstName='Peter' AND LastName='Griffin'");

mysqli_close($con);
?>
```

After the update, the "Persons" table will look like this:

Table 6: The table Persons after UPDATE

FirstName	LastName	Age
Peter	Griffin	36
Glenn	Quagmire	33

6.9 THE DELETE STATEMENT

The DELETE FROM statement is used to delete records from a database table.Syntax:

EXAMPLE 179

```
DELETE FROM table_name
WHERE some_column = some_value
```

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Look at the following "Persons" table:

Table 7: The table Persons

FirstName	LastName	Age
-----------	----------	-----

Peter	Griffin	35
Glenn	Quagmire	33

The following example deletes all the records in the "Persons" table where LastName='Griffin':

EXAMPLE 180

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db")
;
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " .
mysqli_connect_error();
}

mysqli_query($con,"DELETE FROM Persons WHERE
LastName='Griffin'");

mysqli_close($con);
?>
```

After the deletion, the table will look like this:

Table 8: The table Persons after one record is deleted

FirstName	LastName	Age
Glenn	Quagmire	33

7 USER REGISTRATION AND AUTHENTICATION

Almost each web portal contains a login form. After entering user's login name and password and user's authentication, a web portal offers personalized services, like e-mail, data storage, communication tools, etc., to registered users. User's registration and authentication often consist from three parts:

- 1) Acquiring information about an user through some convenient web form, mainly login information – user's name and password. This part can be implemented using just HTML/CSS.
- 2) Processing of the acquired information, e.g., sending and storing it to the database, or username/password validation. This part can be implemented using PHP.
- 3) Storing the information in the database or retrieve some information from the database, like the password corresponding to the given user's name. This part can be implemented using SQL.

As can be seen, the process of user registration and authentication involves all the knowledge we mentioned in the sections before. An implementation of the process will be the most important goal of our effort. The next text shows an example, how the process can be implemented with the knowledge acquired from all the previous text.

7.1 USER REGISTRATION

Technically speaking, the user registration aims to acquire specific information from an user and consequently to store the information in a database. Firstly, it is important to consider, which information we want to acquire from the user. In the example, which serves for an illustration of the whole process, we acquire this information: user's first name, surname, the city, where he/she lives, username and password. Assume that there is already created a database called `registration` on the localhost server. Now, we create the table `users`, which will serve as storage for the information acquired from the user. The table contains 6 fields called: `id`, `first_name`, `surname`, `city`, `username`, `password`. The table is created using the following SQL query:

EXAMPLE 181

```
CREATE TABLE users (  
  id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(20),  
  surname VARCHAR(30),  
  city VARCHAR(20),  
  username VARCHAR(20),  
  password VARCHAR(200)  
)
```

The maximal length of the field `password` is set to 200 to allow for storing an encrypted password into.

7.1.1 HTML – REGISTRATION FORM

Now, we prepare a form we use for acquiring information from the user. It can be created like this:

EXAMPLE 182

```

<form name="registration_form" action="register.php"
method="post">
  <h2>User registration</h2>
  <table>
  <tr>
  <td>First name:</td>
  <td><input type="text" name="first_name" value="" ></td>
  </tr>

  <tr>
  <td>Surname:</td>
  <td><input type="text" name="surname" value="" ></td>
  </tr>

  <tr>
  <td>City:</td>
  <td><input type="text" name="city" value="" ></td>
  </tr>

  <tr>
  <td>Username:</td>
  <td><input type="text" name="username" value="" ></td>
  </tr>

  <tr>
  <td>Password:</td>
  <td><input type="password" name="password" value="" ></td>
  </tr>

  <tr>
  <td></td>
  <td align=right><input type="submit" value="Register"></td>
  </tr>
  </table>
</form>

```

The form (saved as registration_form.html) looks like:

Figure 13: Registration form

User registration

First name:

Surname:

City:

Username:

Password:

The parameter `action` is set to the value “register.php”. After the button Register is pushed, the information from the form is sent to the server and the script is executed. The script process the information. The next section shows, how to create this script.

7.1.2 PHP SCRIPT FOR THE USER’S INFORMATION PROCESSING

The second part of the registration process deals with a processing of data from the previously described form, particularly sending the data to the database. This can be implemented in PHP. Firstly, we store the data from the form to some variables. This can be implemented as follows:

EXAMPLE 183

```
$a=$_POST["first_name"];
$b=$_POST["surname"];
$c=$_POST["city"];
$d=$_POST["username"];
$e=$_POST["password"];
```

Now, we connect to a database. Firstly, we connect to the server with the database. This is made by:

```
mysql_connect("server name", "server user name", "server password");
```

Those parameters are often provided by the webhosting provider. Then we connect to the database:

```
mysql_select_db("database name");
```

The parameter "database name" is also provided by the webhosting provider. In our example, we connect to the localhost server through:

EXAMPLE 184

```
$link = mysql_connect("localhost", "root")
        or die("Unable to connect: " . mysql_error());
print "Connected successfully";
```



```
mysql_select_db("registration") or die("Unable to select  
the database");
```

Now, the data stored in the variables can be passed to the database through a SQL query. Syntax:

```
mysql_query("SQL query");
```

The parameter should contain a SQL query that store the information from the form to an appropriate table.

7.1.3 STORING THE INFORMATION INTO DATABASE

A new record can be inserted into database using the `INSERT INTO` statement. Thus, the information acquired from the user can be inserted into the database using the following query:

EXAMPLE 185

```
INSERT INTO users (first_name, surname, city, username,  
password) VALUES ('$a', '$b', '$c', '$d', '$e')
```

The whole PHP script that processes the information from the form would look like this:

EXAMPLE 186

```
<?php  
$a=$_POST["first_name"];  
$b=$_POST["surname"];  
$c=$_POST["city"];  
$d=$_POST["username"];  
$e = hash("sha512", $_POST["password"]);  
  
$link = mysql_connect("localhost", "root")  
        or die("Unable to connect: " . mysql_error());  
print "Connected successfully";  
mysql_select_db("registration") or die("Unable to select  
the database");  
  
$query = "INSERT INTO users (first_name, surname, city,  
username, password) VALUES ('$a', '$b', '$c', '$d', '$e')";  
mysql_query($query);  
    header("Location: login_form.html");  
    exit();  
?>
```

After the new record is inserted, the user is redirected to “login_form.html”.

7.1.4 PASSWORD ENCRYPTION

To increase the security of our private web pages, it is convenient to encrypt the password before we store it in the database. In this case, we avoid to an unauthorized person, e.g., a database administrator, to find out the passwords in the original form.

So called one-way encryption can be used for the purpose. The encryption transforms a password to a cluster of characters, but without knowing the key for its decryption. Thus, it's impossible to decrypt the encrypted password - this is the main difference between the one-way and a two-way encryption, where there always exist the key corresponding to the cipher and it is possible to get the original information stored in the cipher.

There exist several functions in PHP for these purposes, e.g., `md5()` or `sha1()`, however those algorithm are nowadays considered as obsolete, and its security has been questioned. We describe a more convenient function called `hash()`, which allows to choose some modern encrypting algorithm, e.g., "whirlpool" nebo "sha512".

To implement the encryption in our example, we change the line

```
$e=$_POST["password"];
```

in the way that the password acquired from the user is passed to the variable `$e` in an encrypted form. This can be made, using, e.g., "sha512" with:

```
$e=hash("sha512", $_POST["password"]);
```

Then the password is stored in the encrypted form, for example, when the user fill the password field with 123456, the password is transformed to, for example, ba3253876aed6bc2c6a956df346eab413, using the function `hash("sha512", "123456")`. As the encrypted password is significantly longer than the original password, it is necessary to set up convenient (long enough) length of the corresponding field in the table - in our case we set the length of the password to 200 (see the definition of the table `users`). Later in authentication process, we must compare the password also in the encrypted form (see below in the text).

7.2 USER AUTHENTICATION

Technically speaking, an authentication of the user is based on a comparison between the password acquired from the user in the registration process and the password acquired from the user when the user is logging in. When those two passwords are equal, we allow the user to access to the personalized (private) parts of our web. In other case, the authenticity of the user is not verified and the access to the personalized parts of our web is denied.

7.2.1 LOGIN FORM

A common login form contain two input fields - username and password. In our example, we would implement it using HTML in this way:

EXAMPLE 187

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

```

</head>
<body>
  <form action="login.php" method="post">
    <table>
      <tr><td>Username </td><td><input type="text"
name="username" /></td></tr>
      <tr><td>Password </td><td><input type="password"
name="password" /></td></tr>
      <tr><td colspan="2"><input type="submit" value="...:
Login :... " />
      <input type="reset" value="...: Reset :... "
/></td></tr>
    </table>
  </form>
</body>
</html>

```

Saving it as "login_form.html", it looks in the browser as:

Figure 14: Login form

The parameter `action` is set to the value `login.php`. When the user push the "Login" button, the information given by the user is sent to the server and the script `login.php` is executed.

7.2.2 AUTHENTICATION SCRIPT IN PHP

The authentication script that we use for our example retrieve the information from the form and store it in the variables `$form_username` and `$form_password`. When we use the encrypted form of the password, it is necessary to transform the acquired password to the encrypted form, see the line `$form_password = hash("sha512", $_POST["password"]);` below. An implementation of the authentication can look like this:

EXAMPLE 188

```

<?php
  // set up all settings
  // database settings
  $server = "localhost";
  $user = "root";
  $password = "";

```

7 User registration and authentication

```
$database = "registration";

// registration table settings
$db_table_name = "users";
$db_username_column = "username";
$db_password_column = "password";

// form elements
$form_username = $_POST['username'];
$form_password = hash("sha512", $_POST["password"]); //
password is encrypted

// private section - only for authenticated users
$private_url = "private.php";

//-----
-----
// do not make any changes below

// SQL query that returns password
$sql =
    "SELECT $db_password_column FROM $db_table_name WHERE
$db_username_column = '{ $form_username }'";

echo "Sending this query to DB: <br>".$sql;

$conn = mysql_connect($server, $user, $password)
    or die("Unable to connect: " . mysql_error()); //
server connection
mysql_select_db($database);
// db connection

print "<br>Connected successfully";

$result = mysql_query($sql); //
dotaz na heslo pro uzivatele

// authentication process
if (!$result):
    echo "<br>
    Unable to authenticate the user. Possibly wrong db
    setting. The server haven't returned a response to the
    query.".$sql;
    exit;
endif;
if (!mysql_num_rows($result)):
    echo "<br>User not found.";
    exit;
else:

    $i=0;
```

```
        if (mysql_result($result, $i, $db_password_column) !=
$form_password ):
            echo "<br>Wrong password.";
            exit;
            else:
            endif;
        endif;

        //pokud odpovídají přihlašovací údaje
        session_start();
        header("Cache-control: private");
        //zaregistruje proměnou user_is_logged a nastaví ji na
1
        $_SESSION["user_is_logged"] = 1;
        //a pošle na úvodní soubor chráněné sekce
        header("Location: ".$private_url);
        exit;

?>
```

The first part of the script must be set according to our environment (obtained from the webhosting provider). The first four lines

```
$server = "localhost";
$user = "root";
$password = "";
$database = "registration";
```

must contain the ip address of the server, username/password for the server, and the name of the database that we use. For our example, everything is already set correctly.

The next three lines contain the information about the registration table, i.e., the name of the table, the name of the field, where the usernames are stored and the name of the field, where the passwords are store. For our example, it is:

```
$db_table_name = "users";
$db_username_column = "username";
$db_password_column = "password";
```

The next two lines contain the field names of the login form. For our example, it is:

```
$form_username = $_POST['username'];
$form_password = hash("sha512", $_POST["password"]);
```

Note that the password stored in the variable `$form_password` is in an encrypted form, hence this script would work only with the "encrypted" version of the registration.

The last setting is the name of the page that is displayed after the user is authenticated successfully. In our example, it is:

```
$private_url = "private.php";
```

When the authentication fails, the script shows the message `Wrong password.` or `User not found.`

The rest of the script can be left without any changes. We do not devote to explanation of it, because it is out of the scope of this text. We only note that the script compare the password acquired from the user through the login form with the password stored in the database corresponding to the username given by the user also in the login form.

Now we assure that all the private web pages are accessible only for the authenticated users. For this purpose, we use an additional script saved as "protection.php", which looks like:

EXAMPLE 189

```
<?php
    session_start();
    header("Cache-control: private");
    //if the user is not logged in, refer to login form
    if ($_SESSION["user_is_logged"] != 1){
        header("Location: login_form.html");
        exit();
    }
?>
```

Then, this script is added at the beginning of each private web page using one line in PHP, which has the following form:

EXAMPLE 190

```
require('protection.php');
```

The private web page, e.g., `private.php`, then can be implemented as:

EXAMPLE 191

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta http-equiv="content-type" content="text/html;
charset=windows-1250">
        <meta name="generator" content="PSPad editor,
www.pspad.com">
        <title></title>
    </head>
    <body>
        <?php
            require('protection.php');
        ?>
        Welcome in private section!
    </body>
</html>
```

In the case when an unauthenticated user tries to open the “private.php”, the script “protection.php” redirects the user to “login_form.html”, what protects the private pages to be displayed to this kind of users.

For logging out, we can use:

EXAMPLE 192

```
<?php
    session_start();
    $_SESSION = array();
    session_destroy();
    //Check for if the session is terminated correctly
    if ($_SESSION["user_is_logged"]){
        echo "FATAL ERROR: Cannot terminate session!";
    } else {
        //if everything is ok, refer the user to login form
        //(or somewhere else)
        header("Location: login_form.html");
    }
?>
```

This script log out the logged in user and redirect to login_form.html.

The whole registration and authentication procedure is using so called **sessions**. When using the sessions, some data can be stored to a file on the server and are accessible through the array called `$_SESSION`. It is necessary to call the function `session_start()` on each page before accessing to this array. Due to the weird behavior of a disk cache in MS Internet Explorer, it is recommended to set up own cache administration, what can be done using the statement

```
header("Cache-control: private");
```

Thus, this is also necessary to put this line before any output to HTML.

As soon as the session is started, it is possible to store data into it. It can be done using:

```
$_SESSION['my_variable'] = $value;
```

This statement register (or overwrite) the session variable `$my_variable` and assign it the value `value`, which can be of a different data type. It is of course possible to register more variables. To access to the value, we can use:

```
$value = $_SESSION['my_variable'];
```

It is necessary that there is a variable on the left side. To destroy a variable, we assign it the empty value (“”) or `false`. To destroy the whole session, it is necessary to put at the beginning of a page (but below the function `session_start()`) those two statements:

```
$_SESSION = array(); //clean up all the registered variable
session_destroy(); //close the session
```

Those two lines can be seen in the previously described script “logout.php”. A session is also closed after the browser is closed or after some time period of inactivity.

The whole process of user authentication is proceeds in the following way. Firstly, we redirect a user to “login_form.html”. After the user inputs the correct pair username/password, what is verified through “login.php”, the variable `user_is_logged` in sessions on the server is set to 1. Then, we just verify on all private pages, if the variable `user_is_logged` is set to 1. If so, the user is permitted to display the content of the private pages. In not, when `$_SESSION["user_is_logged"] != 1`, we redirect the user to “login_form.html”.

As can be seen from the previous text, the process of user registration and authentication is not a trivial task, and it uses the knowledge about all the mentioned languages – HTML (CSS), PHP and SQL. On the other side, since almost every web portal implements (offers) the process, we put the main focus on an implementation of the mentioned process, as it will be needed for successful defence of semestral work.

SUMMARY

The topic of web portals is and probably will remain one of the key in computer science. If the reader reads these lines, we can assume that already has a study of all the chapters of this book and is able to create a simple web portal and ensure his administration, which was the main aim of the authors of this book. The prerequisite here is that readers and especially students of the course Portal and its management become familiar with the basic methods of making the structure of web pages using HTML or. XHTML and CSS, basic PHP scripting language, creating and managing MySQL databases and bring all these components into a functional unit - web portal - enabling the presentation of information and communication with the user via web form.

THE LIST OF LITERATURE

- [1] Develop & Host. Php - Apache - Mysql – Windows. Introduction. [Online] Available from <http://www.easyphp.org/introduction.php>
- [2] GILMORE, W. J. Beginning PHP 5 and MySQL: From Novice to Professional. CA United States: Apress, 2005. ISBN 1893115518.
- [3] Home& Learn. Free PHP Tutorial. Create a database with phpMyAdmin. [Online] Available from <http://www.homeandlearn.co.uk/php/php12p2.html>
- [4] HTML Dog. CSS Beginner Tutorial. [Online] Available from <http://www.htmldog.com/guides/css/beginner/>
- [5] VANDERWEEËN, B. Coding a layout in HTML5 and CSS3. [online]. 2012, 2012-02-12 [cit. 2013-02-08]. Available from http://www.bene.be/blog/comments/coding_a_layout_in_html5_and_css3/
- [6] W3Schools.com. CSS Tutorial. [Online] Available from <http://www.w3schools.com/css/>
- [7] W3Schools.com. HTML4 and HTML5 Tutorial. [Online] Available from <http://www.w3schools.com/html/default.asp>