MINISTRY OF INFORMATION TECHNOLOGIES AND COMMUNICATION DEVELOPMENT OF THE REPUBLIC OF UZBEKISTAN

BRANCH CENTER OF RETRAINING AND REQUALIFICATION OF PERSONNEL NEAR THE TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

“Permitted for defense of the thesis”
Director of Network Center A.B.Radjiev
“______”___________ 2016

TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

KADIROV AZIZBEK ABDUMAJITOVICH

Subject: “Cascading Style Sheets in Web-programming”

SUPERVISOR: M.S.YAKUBOV

TASHKENT, 2016

Table of contents

Introduction

Today, more than ever, the role of educational technology in teaching is of great importance because of the use of information and communication technologies. With the help of various applications for distance education, the Internet, teachers, and students themselves, they see the advantage of educational technology. And the web pages design is an important point of creating modern websites, so teaching how to create an impressive and unique design is one of the most important tasks of the teacher.

Since computers are still not widely used in many schools, the teaching process is dominated by traditional methods. It is dominated by the frontal form of work where the teacher had enough interaction with students. Failure to thrive at their own pace and insufficient activity of students was one of the drawbacks of this type of learning. In class, we have children who are not uniform in knowledge and never pay enough attention to those who are not sufficiently mastered the material and those who are above their average. This difference is often hampered by teacher assessment work and how to transfer knowledge to a group of children with different knowledge. The teacher chooses to keep average to good teaching where children with insufficient knowledge would not get the necessary knowledge. The children with insufficient knowledge can progress smoothly without unpleasant feeling of their ignorance, no frustration, and humiliation while for the most advanced children teaching will be boring.

With the development of information and communication technology, especially computers, a number of researchers were trying to see the benefits and effect of their use compared to older traditional learning. For many years, we tried give answers to the question of advantages and disadvantages between traditional and modern teaching where the prevailing educational technology. The period 1967 to 1972 is considered to be a period of consolidation of educational technology, which has become the most commonly used term in the science of pedagogy and the educational process. With the application of educational technology, students can independently progress in mastering teaching materials, choose the pace of work, to repeat the material that is not sufficiently clear, that tests performed immediately get results and track their progress. Interactive, multimedia content provides a great advantage of modern learning over traditional learning. With the application of educational technology we get feedback between the teacher and the student.

What is CSS and why do we use CSS? We use CSS to define styles for your web documents, including the design, layout and variations in display for different devices and screen sizes. You can place your CSS in the <head> of a document with an embedded style sheet, or attach a separate file that defines your styles with an external style sheet. To link an external style sheet to your document, you'll simply add a link to the style sheet in the <head> of the document.

Using CSS, you store the style information in common files that all the pages share. When a user displays a web page, the user's browser loads the style information along with the content of the page.

When a user prints a web page, you can provide different style information that makes the printed page easy to read.

In general, you use HTML to describe the content of the document, not its style. You use CSS to specify the document's style, not its content. Later in this tutorial, you will see some exceptions to this arrangement.

When you use CSS, you normally avoid using HTML features of the markup language so that all your document's style information is in one place.

Therefore, the quality education and the vocational education and the examination oriented education are different, can not completely copy the examination oriented education method. The education method of quality education and vocational education should have its own characteristics.

According to many years' practice, it has been proved that improving the teaching quality of the professional course is the fundamental guarantee for quality education and vocational education.

Purpose of this thesis is to give an overview of educational technologies used in the classroom while teaching Cascading Style Sheets (CSS) on the Web-programming lessons.

To reach this purpose we have **a task** of giving an overview of such technologies as Clustering and Brainstorming Technologies and their usage in teaching Cascading Style Sheets, and the second task is to give an overview of multimedia technologies that can be used in teaching CSS, such as CourseLab.

This subject has a huge **practical relevance** because nowadays Web is developing so fast that many universities are not able to adapt to modern technologies and update their courses. In particular, we have tens of Web scripting frameworks such as JQuery, Prototype, Motools, Angular, React, and every year many new ones appear. In this thesis we will try to overview the solutions of problem of teaching this fast changing science.

The thesis **structure** includes introduction, Chapter 1, Chapter 2, Summary, Reference section and Appendix. Chapter 1 provides information about nowadays state of Web technologies and describes some issues of teaching the Web-development course in universities. Chapter 2 consists of lessons plan and technological maps, lecture texts, assesments, cases and other materials related to the Web development course. In the Summary part we give a conclusion and some recommendations of teaching Web-development course in universities. In the reference section we provide links to the used papers, books and web-sites. Finally, in the Appendix we provide full texts of assesments, tests and other materials.

CHAPTER 1. THEORETICAL ISSUES AND METHODS OF TEACHING MODERN WEB TECHNOLOGIES

1.1. Problems and Issues in learning modern Web technologies

The World Wide Web is a constantly evolving network that has already traveled far beyond its conception in the early 1990s, when it was created to solve a specific problem. State-of-the-art experiments at CERN (the European Laboratory for Particle Physics— now best known as the operator of the Large Hadron Collider) were producing incredible amounts of data—so much that the data was proving unwieldy to distribute to the participating scientists who were spread out across the world.

At this time, the Internet was already in place, with several hundred thousand computers connected to it, so Tim Berners-Lee (a CERN fellow) devised a method of navigating between them using a hyperlinking framework, which came to be known as Hypertext Transfer Protocol, or HTTP. He also created a markup language called HTML, or Hypertext Markup Language. To bring these together, he wrote the first web browser and web server, tools that we now take for granted.

But back then, the concept was revolutionary. The most connectivity so far experienced by at-home modem users was dialing up and connecting to a bulletin board that was hosted by a single computer, where you could communicate and swap data only with other users of that service. Consequently, you needed to be a member of many bulletin board systems in order to effectively communicate electronically with your colleagues and friends.

But Berners-Lee changed all that in one fell swoop, and by the mid-1990s, there were three major graphical web browsers competing for the attention of five million users. It soon became obvious, though, that something was missing. Yes, pages of text and graphics with hyperlinks to take you to other pages was a brilliant concept, but the results didn't reflect the instantaneous potential of computers and the Internet to meet the particular needs of each user with dynamically changing content. Using the Web was a very dry and plain experience, even if we did now have scrolling text and animated GIFs!

Moreover, nowadays, Web is developing so fast, so universities are facing a big problem with teaching so dynamic and fastly changing subject as Web-development. In the last 10 years, we we were witnesses of birth, coming out and dawning of such technologies in Web science, as JQuery and Motools, which were replaced by Angular, React and other technologies. Those students who were learning JQuery, are now stimulated to learn new technologies. Another problem is the shortage of actual and high-quality manuals for new technologies. If you want to be well informed in new flows of Web, you should read user manuals, forums and documentations, and it takes time to pick and convert such distributed and sliced information into quality textbook. In addition, many institutions work with the inertia of thinking, and it is quite difficult to change a training program and add new subjects. Therefore, the task and responsibility of universities and teachers is to build basic fundamental knowledges and to develop a self-studying ability in students, so they can learn new technologies on their own. Therefore, one of the main tasks described in this thesis – developing all-around, creative, self-maintained personality, and for that, we would use modern educational technologies, such as brainstorming, clustering and case study.

With the development of information and communication technology, especially computers, a number of researchers (Morrison et al., 2010) were trying see the benefits and the effect of their use compared to older traditional learning. many years, we tried to give answers to the question of advantages and disadvantages between traditional and modern teaching where the prevailing educational technology. The period from 1967. to 1972. is considered to be a of consolidation of educational technology, which has become the most commonly used term in the science of pedagogy and the educational process (Даниловић, 2004). With the application of educational technology, students can independently progress in mastering teaching materials, to choose the pace of work, to repeat the material that is not sufficiently clear, that after tests performed immediately get results and track their progress. Interactive, multimedia content provides a great advantage of modern learning over traditional learning. With the application of educational technology we get feedback between the teacher and the student.

Among the first studies on the comparison of the traditional and modern with the help of educational technologies research was Clark Richard (Clark, R. 1983). He tried to compare research between lectures and computer guidance and instruction to determine which the better way of learning is. He came to the conclusion that they are both effective depending on the ways they are used. The same conclusion came by other authors (Dynarski et al. 2007; Kulik, 2003) and is that there are some major differences in the use of educational technology and traditional teaching. On the other hand, research at the Center for Educational Research in Pittsburgh within Individually Prescribed Instruction showed that computers are better tailored to the individual abilities of students, rather than teachers themselves. Educational technology must inevitably be integrated into classrooms and curricula (Clements and Sarama, 2003; Glaubke 2007; NAEYC Fred Rogers Center, 2012). With the advent of educational technology in the classroom teacher, education is faced with the challenge that teachers integrate educational technology in their daily work. Numerous studies have shown that a small number of teachers is willing to integrate educational technology in their teaching activities (Becker, 2000; Hermans et al., 2008; Stošić and Stošić 2013; Wang et al., 2004). The reason is that there are two categories of teachers in the understanding of educational technology. Some of them have thorough understanding of modern technical appliances and their operation while others it is necessary for them to gain additional technical knowledge of the appliances methods, teaching methods, student-teacher relationship... These two groups represent a group of teachers between older and younger teachers. Older teachers during their study did not have the possibility of training with modern technical appliances, did not have the information technology, educational technology... the younger generation of teachers possess the knowledge required for the use of educational technology. For a better understanding of educational technology requires a set of computer science, pedagogy, psychology, cybernetics, informatics... The knowledge teachers possess is sufficient for a basic use of education technology. However, educational technology is one big system. First of all, teachers have a basic knowledge of the use of educational technology. It takes more professional training through a variety of conferences, courses, professional literature, seminars... in order to get a better knowledge in the use of educational technology. The fact is that under use of educational technology, primarily due to poor school equipment necessary resources, insufficient information and of teachers and the lack of interest and lack of motivation of teachers to use them. Teachers have to be motivated to use the same because the use of educational technology in teaching provides better interaction with students, better reception of information because the students receive knowledge visual, auditory and way. Among other things, an educational technology motivates students to work independently where the student is more motivated to return to learning and because modern technical equipment is widely available at any given moment.

1.2. Brainstorming

Brainstorming provides a nearly guaranteed solution to writer's block. It's actually a very easy process.

1) Begin with a blank piece of paper or computer screen.

2) Write your paper's topic, such as "The Ethics of Cigarette Advertising," at the top.

3) Write down everything you can about the topic; omit nothing, no matter how bizarre, and don't stop until you are completely out of ideas. Don't worry about grammar or editing. Here's an example:

Paper Topic: Making colored text on Web-pages

a) Use <font> tag

b) Use text attribute of <body>

c) Use colored images instead of text

d) Use CSS

4) Look at the list above and reconsider the paper's topic. Ideas (d) and (i) stray from the topic, so cut them.

5) Organize the remaining points. Idea (a) provides a decent place to start a draft, since it touches on a central truth about smoking.

6) Next, try to arrange the other pointsin the order in which they would logically appear in your essay. It helps to think about patterns into which ideas would fall, such as "Appeal of Ads," "Limits on Ads," and "Future of Ads." You could write down these categories and then categorize your ideas from the brainstorming list. You'll end up with a working outline for the paper.

7) New ideas may occur to you as you organize the material. That's okay as long as these ideas relate to the topic.

8) You're almost ready to begin a draft, or at least an introductory paragraph

So, the brainstorming is very useful in the beginning of the lesson to start generating ideas and to focus the students' attention to the topic. We will use brainstorming and other educational technologies in creating the lesson plan in the next chapter.

1.3. Clustering

Clustering is a type of prewriting that allows you to explore many ideas as soon as they occur to you. Like brainstorming or free associating, clustering allows you to begin without clear ideas.

To begin to cluster, choose a word that is central to your assignment. For example, if you were writing a paper about the value of a college education, you might choose the word "expectations" and write that word in the middle of your sheet of paper. Circle "expectations," then write words all around it --words that occur to you as you think of "expectations." Write down all words that you associate with "expectations," words that at first may seem to be random. Write quickly, circling each word, grouping words around the central word. Connect your new words to previous ones with lines; when you feel you have exhausted a particular avenue of associations, go back to your central word and begin again.

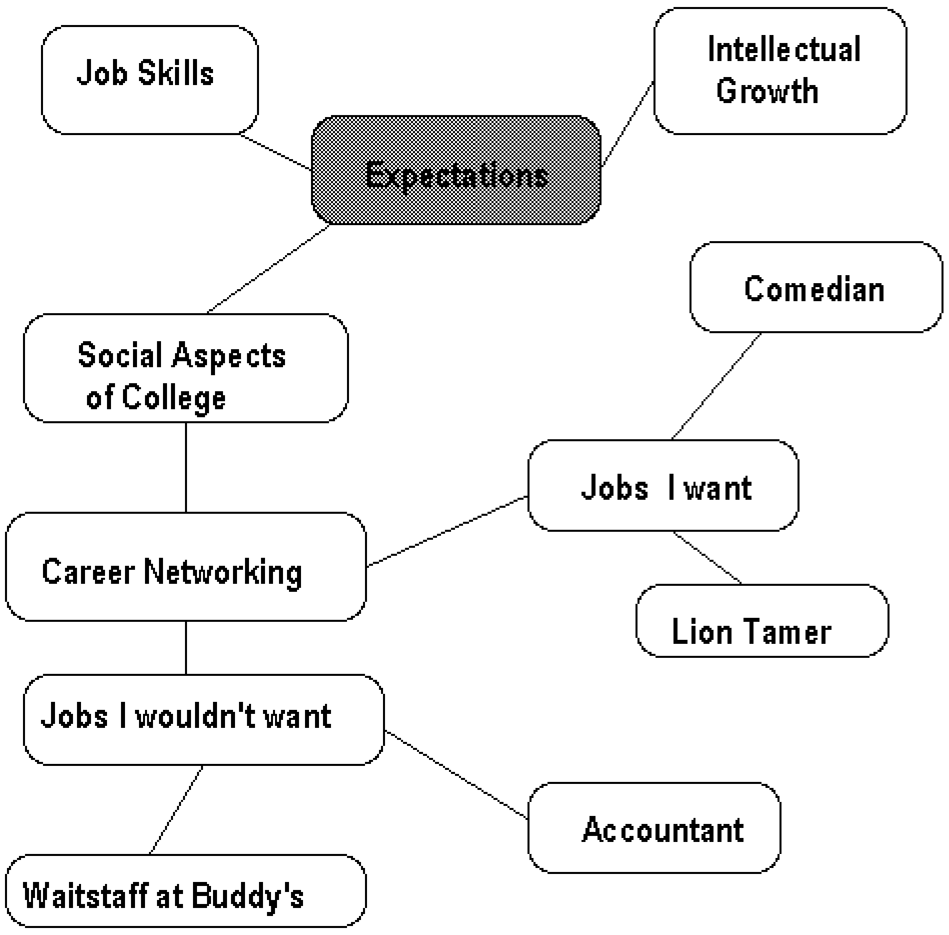


Fig.1-1. Cluster example

For example, "expectations" might lead you to consider "the social aspects of college," which may lead you to consider "career networking." You may then find yourself writing down words that compare the types of jobs you might get through career networking. You may end up asking yourself questions such as "What sorts of jobs do I want? Not want?" Have fun with this exercise; even silly questions can open avenues to explore, such as "What if I ended up waiting tables at Buddy's?" "Would I rather be a lion-tamer or an accountant?" "What about my brilliant career as a stand-up comedian?"

Some words will take you nowhere; with other words you may discover that you have many related words to write. Random associations eventually become patterns of logic as you look over your work. After looking over the clustering exercise above, you might conclude that you want an exciting career as a performer of some type rather than a job in the service sector or behind a desk.

Now your sample paper about the value of a college education has some focus: how you expect college to lead to an interesting career that involves creativity, skill, and performance. You might then want to return to the phrase "Job Skills" and develop that part of your cluster, noting the skills that you'd need to reach your ideal career.

Clustering does not take the place of a linear, traditional outline; but, as the example shows, it allows you to explore ideas before committing them to a particular order.

1.4. Case study

A case study is an account of an activity, event or problem that contains a real or hypothetical situation and includes the complexities you would encounter in the workplace. Case studies are used to help you see how the complexities of real life influence decisions.

Analysing a case study requires you to practice applying your knowledge and your thinking skills to a real situation. To learn from a case study analysis you will be "analysing, applying knowledge, reasoning and drawing conclusions" (Kardos & Smith 1979).

According to Kardos and Smith (1979) a good case has the following features:

1. It is taken from real life (true identities may be concealed).
2. It consists of many parts and each part usually ends with problems and points for discussion. There may not be a clear cut off point to the situation.
3. It includes sufficient information for the reader to treat problems and issues.
4. It is believable for the reader (the case contains the setting, personalities, sequence of events, problems and conflicts)

Types of case study

Your course may include all the information you require for the case study and in this case all students would be analysing the same case study. This may take the form of an historical case study where you analyse the causes and consequences of a situation and discuss the lessons learned. You are essentially outside the situation.

Other types of case studies require you to imagine or role play that you are in the situation and to make plausible recommendations to senior management or ministers. Some case studies require you to solve a problem by developing a new design. These types of case studies are problem orientated.

Alternatively you may be able to choose a real situation, such as an event in your workplace, to analyse as a case study, either as a problem orientated situation or an historical case /situation. In this instance, you would need to locate the information necessary to write a clear description of the case before you can analyse the situation and make recommendations.

Some examples of case study assignments are:

Historical case study

Take a recent company collapse (eg HIH, FROGGY, ENRON) and analyse what went wrong.

Problem orientated case study

Using cost benefit risk analysis, determine the current and future market opportunity of company X in country Y.

Case Study Report

A case study analysis is usually presented as a report and will therefore contain many of the features and structure of reports in general. This section will briefly describe each section, its purpose and structure.

A. Title page

The title page presents routine information and hints at the report's content through an informative title. Design your title page to be simple yet functional and appropriate for your audience. Common elements to include on the title page include:

- Your Institution's name
- Title of the report
- Author/s (include student number if appropriate)

▪Name of person or group to whom you submit the report

▪Course name (or department/group or committee name)

▪Date of submission

B. Executive summary

The executive summary is usually read by senior management. The manager will use the information in the executive summary to decide what action to take and who will carry it out. An executive summary should include an overview of the whole report and is longer than an abstract for a professional journal. It can be from one to a couple of pages, but try to keep it under 2 pages if possible. Headings can be used but there is no need to number these. In your own words present clearly and briefly:

▪the topic area of the report

▪the report's primary aim/s

▪state what was achieved (key finding)

▪a summary of your approach

▪significant findings

▪a summary of the report's recommendations

C. Contents page

Readers can use this to get a sense of how the report is structured and can skim the contents page for relevant sections to read. Include heading, subheading and page numbers. Usually in large reports a decimal numbering system for headings and subheadings are used. If it is a large report with many tables and figures in the body, a list of figures and a list of terminology or symbols can be included after the contents page.

D. Introduction

The introduction is very important as it sets the context for the report. Summarise the brief (your task), briefly outline the case and focus on its significance for the reader, state the report's aim(s) and describe how the report is organised. Readers use the introduction to locate the aim of your report and to decide which sections of the report they need to read. While you may include the key problem you have identified and its significance, it is not usual to detail findings or recommendations in the introduction.

E. Case study report body

The previous sections (title page, executive summary, contents, tables of figures, introduction) are preliminary sections.

It is difficult to give a single precise description of how a case study report should be organised as many models and variations exist. Organisation will depend on the type of report (eg; design, management), the type of case study investigation (eg; historical, problem orientated), and even the discipline or field you are writing in. Ultimately, the writer decides how best to organise and explain the case, the methodology and the recommendations. The following descriptions are examples only and are drawn from the field of risk management.

1. Historical case study

An historical case study's body sections may be organised as follows:

▪**Context** — Describe the case or situation being investigated. Focus on the facts of the situation.

▪**Approach** - Use topic based headings and a chronological sequence to give a summary and discussion of contributing factors (usually focusing on a specific time period in the past) that lead to and resulted from the situation described in the case study. Refer to theories, relevant publications or prior cases to explain and justify your interpretations of the situation. Problems and solutions and previous recommendations that were made are highlighted and briefly commented upon (eg; which problems were eventually solved and how they were solved, or which problems continued and why they remained unsolved).

▪**Conclusion**- Try to answer the following questions. What else has been achieved since the situation occurred? Have all recommendations been implemented? What may happen in the future?

An example of an historical case study report can be found at: http://www.colorado.edu/hazards/wp/wp98.html it is titled: "A Case Study of Florida's Emergency Management Since Hurricane Andrew". This case study is one of a series of research working papers for the Natural Hazards Research and Application Information Centre, Institute of Behavioural Science, University of Colorado. Additional historical case studies with slightly different body sections can be found at this site: http://www.colorado.edu/hazards/WP

2. Problem-orientated case study

A problem orientated case study's body sections may be organised as follows:

Headings should be informative and descriptive providing a clue to the contents of the section.

▪Describe the context of the case. Present the central issue you will be analysing, what decisions have already been made, what communication processes are occurring in the situation. Focus on the facts.

▪Explain your methodology. Identify problems that are demonstrated in the case (use visuals if appropriate) and also explain and justify your choice of analysis tools (eg SWOT, PEST, Force Field…),

▪Present summaries of your findings (put details in the appendices) and indicate how you decide what is acceptable/not acceptable as a solution.

▪Present an action plan for the recommendations. Recommendations in a case study report should be fairly detailed. Include an action plan that details who should take action, when and how (eg; specifications, steps to follow), and how to assess the action taken. For example, in a case study report you may decide the likelihood of 3 scenarios pose the greatest risks for your company but each poses a risk in unique ways. For each scenario clearly state who is responsible, what action they should take and how they can assess the recommendation.

F. Conclusions

Every report should include a concluding statement/s on the subject of the report. Restate the aim of the report and state how you have achieved it. Present the main findings and key recommendations in a summarised form for the reader's benefit. You should also restate the limitations of the report.

G. Appendices

Appendices provide additional or supporting information that while not essential to understanding the main facts and recommendations, may be of interest to the expert reader and are evidence of your research and analysis. Appendices can be tables of raw data, detailed calculations, design drawings, maps, copies of a questionnaire or survey etc. Appendices are normally listed as Appendix A, Appendix B, Appendix C, and so forth. Give each appendix a clear informative title. Appendices and reference lists are supplementary sections of a report.

H. Reference list

This is a list of all the sources of information you have referred to in the report. Many schools in engineering recommend the author date system. It's recommended to check with your course facilitators on their preferences.

1.5. Creating multimedia presentations with CourseLab

CourseLab "is a easy-to-use, e-learning authoring system that offers a programming-free environment for creating high-quality interactive e-learning courses which can be published on the Internet, Learning Management Systems (LMS), CD-ROMS and other devices"(CourseLab, 2012).

Brief History: CourseLab was developed by WebSoft company located in Moscow, Russia. WebSoft was formed in 1999 by the group of Moscow State University graduates. Since then one of the main company goals is to develop the powerful, yet easy-to-use eLearning tools.

This company has release various version of commercial software. In Feb 2008, this company released basic version of eLearning content creation tools called as CourseLab as a freeware. The term "freeware" means that you can install and use it for free. Some advanced options, such as import of PPT slides, are made available by paying for extra amounts. But CourseLab is not open code software; therefore you cannot make any changes in the CourseLab software.

Feature of CourseLab Tool

CourseLab features are specially designed to increase e-learning content creation productivity. These features are given below: e-Learning standards:

Learning modules created using CourseLab are compliant with the following e-Learning standards:

- AICC (http://www.aicc.org/) •
- SCORM 1.2 (http://www.adlnet.org/)
- SCORM 2004 (SCORM 1.3) (http://www.adlnet.org/)

Ready-to-use e-learning module template: CourseLab comes with the large set of ready-to-use e-learning module templates. Just select the template that is suitable for your needs to start new module. Templates are easily modifiable and modified templates can be saved to templates library.

WYSIWYG environment: Course Lab supports WYSIWYG (What You See Is What You Get) environment. so no HTML or other programming skills required for content creations.

Large object library: CourseLab contains the large library of ready-to-use complex objects, that covers most of frequently used e-learning needs. Objects are highly customizable to fit virtually any design – just add object on the slide and change its parameters.

Inheritance of objects: You can share once inserted CourseLab object through multiple slides or even entire e-learning module. This feature allows to save development time and efforts drastically.

Simple PowerPoint-like user interface: Familiar PowerPoint-like user interface allows easily get into the way of creating e-learning content.

Import PowerPoint presentation into the module: If you already have PowerPoint presentations then you can easily import it into the e-learning module to save development time. Objects of imported presentation will be translated to CourseLab objects and can be edited further in CourseLab.

Simple wizard-based publication: Instantly publish content to multiple platforms with the few mouse clicks. Content package will be created automatically depending on selected publish option.

CHAPTER 2. TEACHING CSS USING MODERN EDUCATION TECHNOLOGIES

2.1 Contents of study program in Web development

SYLLABUS

for the subject of «Web development»

to 2016/2017 academic year

Brief description of subject								
Location and name of HEO(High Education Organization):	Nukus branch of the Tashkent University of Information Technologies				A.Dosnazarov street, 74, Nukus			
Department:	Programming Engineering				Faculty of Computer Engineering			
Specialization:	5330500 - Computer Education engineering							
Information about professor of the subject:	Assistant teacher Azizbek Kadirov			**e-mail:**	a.kadirov@umail.uz			
Room № and schedule of the subject:	103			**Duration of Subject:**	02.09.2016-20.01.2017			
Working in Individual graphics:	Monday, Friday 14.00 to 16.00							
Distributed time for subject	**Auditory class**							
	Lecture:	36	**Practice**	36	**Lab**	**36**	**Independent Education:**	25
Relationship with other subjects (Prerequisites):	C++ programming, Introduction to algorithms, Internet basics, Informatics, Programming languages, Programming technologies							
Content of Subject								
Actuality and brief content	**Actuality of subject:** Today information technologies are widely integrated to all fields, in particular to education field. In all governmental institutions there are a lot of automatization works held. Every organization should have its own							

of subject:	website. A lot of online shops, maps, museums are created every day. That's why requirements to web application development are becoming higher and higher. Every month new web application technologies are created. This is the reason of this topic being actual. **The content of subject:** "Web development" – this course introduces students with main technologies used in web development, in particular, HTML, CSS, JavaScript, PHP and MySQL. First three chapters cover the client-side web applications development, and the last two ones cover server-side development.
Requirements for students	Specifically students should be familiar with programming basics and working with Internet global network.. - should be polite with teachers and coursemates; - should follow all the rules of university; - shouldn't use mobile phone during the; - should finish and apply all the given homeworks and independence work in time; - It's banned to copy from someone(plague); - should not to miss the lesson!

Week	Lecture Topics	Hours
1.	Introduction to web. Basic terms and principles	2
2.	HTML: Document structure, basic tags	2
3.	HTML: Lists, Graphics and Links	2
4.	HTML: Tables	2
5.	HTML: Forms	2
6.	Cascading Style Sheets: working principles	4
7.	Java Sciprt basics	2
8.	Introduction to PHP programming. Language syntax.	2
9.	Variables and Constants in PHP. $_POST and $_GET global arrays. Conditionals	2
10.	Working with strings. Strings functions	2
11.	Arrays and loops in PHP. Foreach loop	2
12.	MySQL Databases. Table and data types. SQL syntax	2
13.	Connecting to MySQL through PHP	4
14.	PHP frameworks	2

15.	Content Management Systems	4

Practical lessons topics

1	Creating sample HTMLdocument	2
2	Working with Dreamweaver editor	2
3	Designing HTML pages: graphics, links, lists	2
4	Working with tables	2
5	Creating forms in HTML	2
6	Designing pages with the help of CSS in Dreamweaver	2
7	Working with JavaScript	2
8	Working with Denwer. Sample PHP script	2
9	Learning PHP synax. Working with $_GET and $_POST	2
10	Working with string functions in PHP	2
11	Learning arrays and loops in PHP	2
12	Working with phpMyAdmin and creating databases and tables	2
13	SQL basics. Working with INSERT and UPDATE queries	2
14	SQL basics. Working with SELECT query	2
15	Connecting to databases in PHP	2
16	Installing the CodeIgniter framework	4
17	Installing Joomla! Content Management System	2

<u>Evaluation system of students:</u>

/p	**Name of controlling tasks**	**Maximum ball**	**Distribution of current control(CC) and Interim control(IC)**	
	I. Distribution of current control	**36 ball**	18	18
	Lecture and practice works	Maximum ball	*1-CC*	*2-CC*
l.	Attendance of student to practice. Activities in Practice.	4	0-2	0-2
?.	Finishing siccessfully all practice and laboratory tasks	24	0-12	0-12
3.	Control works, Writing tasks and test results	8	0-4	0-4

II. Interim control	**34 ball**		
1. Attendance and Activities in lecture.	4	0-2	0-2
2. Tests or exams	22	0-10	0-12
3. Assesments, independent work	8	0-4	0-4
III. Final exam	**30 ball**	0-30	
Total:	**100 ball**		

Refere nces:

1. Robin Nixon. Learning PHP, MySQL, JavaScript, CSS&HTML
2. Д.Котерев, А.Котерев. PHP5. наиболее полное руководство./Санкт Петербург «БХВ-Петербург» - 2005.- 1120 с.
3. Уилтон П. JAVASCRIPT. Основы. Символ-плюс. 2002. 1056 с.
4. Кингли-Хью Э., Кингли-Хью К. JAVASCRIPT 1.5: Учебный курс. Питер. 1-е издание. 2002.
5. Дронов В. JavaScript в Web дизайне. Питер. 2001.

2.2. Education technology for CSS lesson.

Subject	Cascading Style Sheets
Number of students	20
Time	2 hours
Lesson type	Lecture
Lesson purpose: give an information about CSS and learn how to work with them	
Discussion questions	1. What kind of CSS exists? 2. When we use CSS? 3. Advantages of CSS
Pedagogical tasks: Teach the types of CSS, using CSS in a proper way, teach advantages and disadvantages of CSS.	**Education activity results:** Students learn CSS syntax and learn to use CSS rules in a proper way.
Teaching methods	Brainstorming, clustering, case study
Tools	Tutorials, PC, Projector, presentations, handouts, lecture texts.
Education form	Individual work
Monitoring and evaluation	Assessment

2.3. Technological map for the lesson

Stages	**Activity**	
	Teacher	**Students**

1-stage. **Beginning of the lesson (5 min.)**	1.1. Tells about the web-pages designing problem	Listening to the teacher
2-stage. **Brainstorming (15 min.)**	2.1. Suggests to find a solution of web-page layout design problem	Offer their solutions (based on tables, indents etc)
3-stage. **Lecture** **(45 min.)**	3.1.Tells about CSS and its syntax, usage of CSS. 3.2. Gives an example of solving a problem from the previous stage	Writing the lecture, ask questions, discuss solution
4-stage. **Consolidation (5 min.)**	4.1. Suggests to draw a cluster on the subject "CSS"	Draw a cluster based on new received information
5-stage. Summary (10 min)	5.1. Gives an assessment to check the knowledge	Students fill in the assessment form until the end of the lesson

2.4. The lecture text

Introduction to CSS

Using *CSS (Cascading Style Sheets),* you can apply styles to your web pages to make them look exactly how you want. This works because CSS is connected to the DOM (Document Object Model), which I explained in Chapter 14.

With CSS and its integration with the DOM, you can quickly and easily restyle any element. For example, if you don't like the default look of the <h1>, <h2>, and other heading tags, you can assign new styles to override the default settings for the font family and size used, or whether bold or italics should be set, and many more properties too.

One way you can add styling to a web page is by inserting the required statements into the head of a web page between the <head> and </head> tags. So, to change the style of the <h1> tag, you might use the following code (I'll explain the syntax later):

```
<styte>
H1 { color:red; font-size:3em; font-family:Arial; }
</styte>
```

Within an HTML page this might look like Example 19-1 (see Figure 2-1), which, like all the examples in this chapter, uses the standard HTML5 DOCTYPE declaration.

Example 19-1. A *simple HTML page*

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World</title>
<style>
h1 { color:red; font-size:3em; font-family:Arial; }
</style>
</head>
<body>
```

```
<h1>Hello there</h1>
</body>
</html>
```

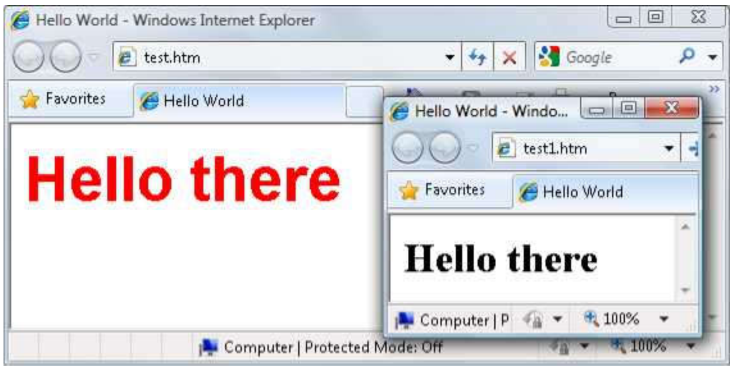


Figure 2-1. Styling a tag, with the original style shown in the inset

Importing a Style Sheet

When you wish to style a whole site, rather than a single page, a better way to manage style sheets is to move them completely out of your web pages to separate files, and then import the ones you need. This lets you apply different style sheets for different layouts (such as web and print), without changing the HTMLThere are a couple of different ways you can achieve this, the first of which is by using the CSS @import directive like this:

```
<styte>
 @import url('styles.css');
</styte>
```

This statement tells the browser to fetch a style sheet with the name *styles.css*. The @import command is quite flexible in that you can create style sheets that themselves pull in other style sheets, and so on. You need to just make sure that there are no <style> or </style> tags in any of your external style sheets, or they will not work.

Importing CSS from Within HTML

You can also include a style sheet with the HTML <link> tag like this:

```
<link rel='stylesheet' type='text/css'
href='styles.css'>
```

This has the exact same effect as the @import directive, except that <link> is an HTML- only tag and is not a valid style directive, so it cannot be used from within one style sheet to pull in another, and also cannot be placed within a pair of <style> ... </ style> tags.

Just as you can use multiple @import directives within your CSS to include multiple external style sheets, you can also use as many <link> elements as you like in your HTML.

Embedded Style Settings

There's also nothing stopping you from individually setting or overriding certain styles for the current page on a case-by-case basis by inserting style declarations directly within HTML, like this (which results in italic, blue text within the tags):

```
<div style='font-style:italic; color:blue;'>Hello
there</div>
```

But this should be reserved only for the most exceptional circumstances, as it breaks the separation of content and presentation.

Using IDs

A better solution for setting the style of an element is to assign an ID to it in the HTML, like this:

```
<div id='welcome'>Hello there</div>
```

This states that the contents of the <div> with the ID welcome should have applied to them the style defined in the welcome style setting. The matching CSS statement for this might look like the following

```
#welcome { font-style:italic; color:blue; }
```

Note the use of the # symbol, which specifies that only the ID with the name welcome should be styled with this statement.

Using Classes

If you would like to apply the same style to many elements, you do not have to give each one a different ID because you can specify a class to manage them all, like this:

```
<div class='welcome'>Hello</div>
```

This states that the contents of this element (and any others that use the class) should have applied to them the style defined in the welcome class. Once a class is applied you can use the following rule, either in the page header or within an external style sheet for setting the styles for the class: `.welcome { font-style:italic; color:blue; }`

Instead of the # symbol, which is reserved for IDs, class statements are prefaced with a . (period).

Using Semicolons

In CSS, semicolons are used to separate multiple CSS statements on the same line. But if there is only one statement in a rule (or in an inline style setting within an HTML tag), you can omit the semicolon, as you can for the final statement in a group.

However, to avoid hard-to-find CSS errors, you may prefer to always use a semicolon after every CSS setting. You can then copy and paste them, and otherwise modify properties, without worrying about removing semicolons where they aren't strictly necessary or having to add them where they are required.

CSS Rules

Each statement in a CSS rule starts with a selector, which is the item to which the rule will be applied. For example, in this assignment, hi is the selector being given a font size 240% larger than the default:

```
H1 { font-size:240%; }
```

font-size is a property. Providing a value of 240% to the font-size property of the selector ensures that the contents of all <hi> ... </hi> pairs of tags will be displayed at a font size that is 240% of the default size. All changes in rules must be within the { and } symbols that follow the selector. In font-size:240%; the part before the : (colon) is the property, while the remainder is the value applied to it.

Last comes a ; (semicolon) to end the statement. In this instance, because font-size is the last property in the rule, the semicolon is not required (but it would be if another assignment were to follow).

Multiple Assignments

You can create multiple style declarations in a couple of different ways. First, you can concatenate them on the same line, like this:

```
h1 { font-size:240%; color:btue; }
```

This adds a second assignment that changes the color of all <h1> headings to blue. You can also place the assignments one per line, like the following:

```
h1 { font-size:240%; cotor:btue; }
```

Or you can space out the assignments a little more, so that they line up below each other in a column at the colons, like this:

```
h1 {
font-size:240%; color :blue;
}
```

This way, you can easily see where each new set of rules begins, because the selector is always in the first column, and the assignments that follow are neatly lined up with all property values starting at the same horizontal offset. In the preceding examples, the final semicolon is unnecessary, but should you ever want to concatenate any such groups of statements into a single line, it is very quick to do with all semicolons already in place.

You can specify the same selector as many times as you want, and CSS combines all the properties. So the previous example could also be specified as:

```
H1 { font-size: 240%; } hi { color : blue; }
```

There is no right or wrong way to lay out your CSS, but I recommend that you at least try to keep each block of CSS consistent with itself, so that other people can take it in at a glance.

What if you specified the same property to the same selector twice?

```
H1 { color : red; } hi { color : blue; }
```

The last value specified—in this case, blue—would apply. In a single file, repeating the same property for the same selector would be pointless, but such repetition happens frequently in real-life web pages when multiple style sheets are applied. It's one of the valuable features of CSS, and where the term cascading comes from.

Using Comments

It is a good idea to comment your CSS rules, even if you describe only the main groups of statements rather than all or most of them. You can do this in two different ways. First, you can place a comment within a pair of /* ... */ tags, like this:

```
/* This is a CSS comment */
```

Or you can extend a comment over many lines, like this:

```
/*
A Multiline
comment
*/
```

When using multiline comments, note that you cannot nest singleline (or any other) comments within them. Doing so can lead to unpredictable errors.

Style Types

There are a number of different style types, ranging from the default styles set up by your browser (and any user styles you may have applied in your browser to override its defaults), through inline or embedded styles, to external style sheets. The styles defined in each type have a hierarchy of precedence, from low to high.

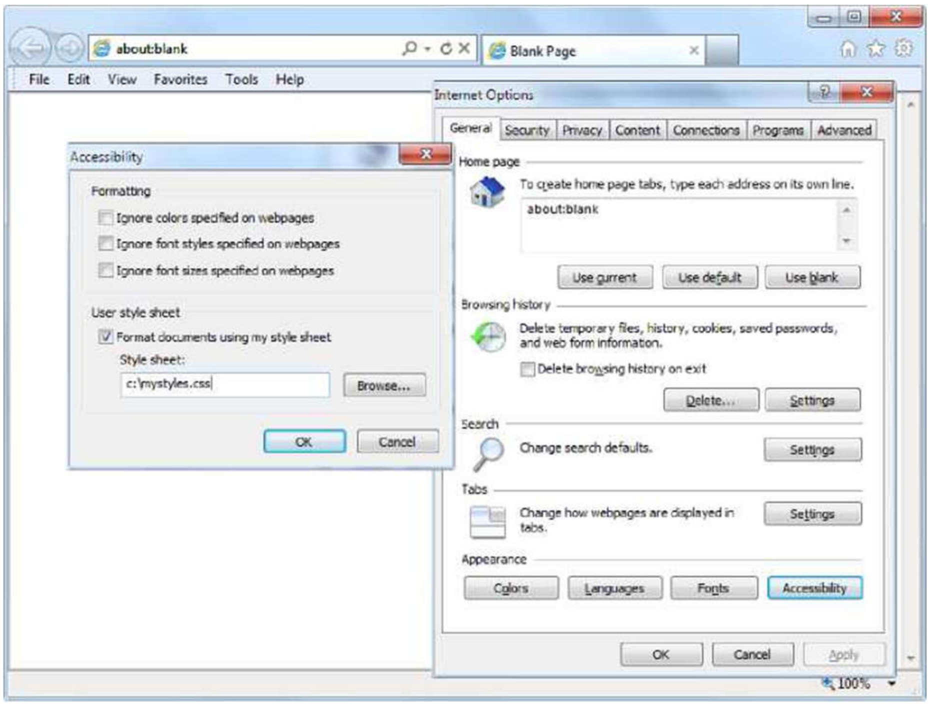
Default Styles

The lowest level of style precedence is the default styling applied by a web browser. These styles are created as a fallback for when a web page doesn’t have any styles, and they are intended to be a generic set of styles will display reasonably well in most instances.

Pre-CSS, these were the only styles applied to a document, and only a handful of them could be changed by a web page (such as font face, color, and size, and a few element sizing arguments).

User Styles

These are the next highest precedence of styles, and they are supported by most modern browsers but are implemented differently by each. If you would like to learn how to create your own default styles for browsing, use a search engine to enter your browser name followed by “user styles” (e.g., “Firefox user styles” or “Opera user styles”) to find out how. Figure 2-2 shows a user style sheet being applied to Microsoft Internet Explorer. If a user style is assigned that has already been defined as a browser default, it will then override the browser’s default setting. Any styles not defined in a user style sheet will retain their default values as set up in the browser.


about:blank
Blank Page
File Edit View Favorites Tools Help
Internet Options
General Security Privacy Content Connections Programs Advanced
Accessibility
OK Cancel

Accessibility

Formatting
- [ ] Ignore colors specified on webpages
- [ ] Ignore font styles specified on webpages
- [ ] Ignore font sizes specified on webpages

User style sheet
- [✓] Format documents using my style sheet

Style sheet: c:\mystyles.css

Internet Options – General

Home page: To create home page tabs, type each address on its own line. about:blank

Browsing history: Delete temporary files, history, cookies, saved passwords, and web form information.
- [ ] Delete browsing history on exit

Search: Change search defaults.

Tabs: Change how webpages are displayed in tabs.

Appearance: Colors, Languages, Fonts, Accessibility

Figure 2-2. Applying a user style to Internet Explorer

External Style Sheets

The next types of styles are those assigned in an external style sheet. These settings will override any assigned either by the user or by the browser. External style sheets are the recommended way to create your styles because you can produce different style sheets for different purposes such as styling for general web use, for viewing on a mobile browser with a smaller screen, for printing purposes, and so on. Just apply the one needed for each type of media when you create the web page.

Internal Styles

Then there are internal styles, which you create within <style> ... </style> tags, and which take precedence over all the preceding style types. At this point, though, you are beginning to break the separation between styling and content, as any external style sheets loaded in at the same time will have a lower precedence.

Inline Styles

Finally, inline styles are where you assign a property directly to an element. They have the highest precedence of any style type, and are used like this:

```
    <a href=http://google.com style="color:green;">
Visit Googte </a>
```

In this example, the link specified will be displayed in green, regardless of any default or other color settings applied by any other type of style sheet, whether directly to this link or generically for all links.

When you use this type of styling, you are breaking the separation between layout and content; therefore, it is recommended that you do so only when you have a very good reason.

CSS Selectors

The means by which you access one or more elements is called selection, and the part of a CSS rule that does this is known as a selector. As you might expect, there are many varieties of selector.

The Type Selector

The type selector works on types of HTML elements such as <p> or <i>. For example, the following rule will ensure that all text within <p> ... </p> tags is fully justified:

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

The Descendant Selector

Descendant selectors let you apply styles to elements that are contained within other elements. For example, the following rule sets all text within <b> ... </b> tags to red, but only if they occur within <p> ... </p> tags (like this: <p><b>Hello</b> there</p>):

```
p b { color:red; }
```

Descendant selectors can continue nesting indefinitely, so the following is a perfectly valid rule to make the text blue within bold text, inside a list element of an unordered list:

```
ul li b { color:blue; }
```

As a practical example, suppose you want to use a different numbering system for an ordered list that is nested within another ordered list. You can achieve this in the following way, which will replace the default numeric numbering (starting from 1) with lowercase letters (starting from a):

```
<!DOCTYPE html>
<html>
<head>
<style>
ol ol { list-style-type:lower-alpha; }
</style>
</head>
<body>
<ol>
<li>One</li>
<li>Two</li>
<li>Three
```

```
<ol>
<li>One</li>
<li>Two</li>
<li>Three</li>
</ol>
</li>
</ol>
</body>
</html>
```

The result of loading this HTML into a web browser is as follows, in which you can see that the second list elements display differently:

```
1.One
2.Two
3.Three
  a.One
  b.Two
  c.Three
```

The Child Selector

The child selector is similar to the descendant selector but is more restrictive about when the style will be applied, by selecting only those elements that are direct children of another element. For example, the following code uses a descendant selector that will change any bold text within a paragraph to red, even if the bold text is itself within italics (like this <p><i><b>Hello</b> there</i></p>):

```
p b { color:red; }
```

In this instance, the word Hello displays in red. However, when this more general type of behavior is not required, a child selector can be used to narrow the scope of the selector. For example, the following child selector will set bold text to red only if the element is a direct child of a paragraph, and is not itself contained within another element:

```
p > b { color:red; }
```

Now the word Hello will not change color because it is not a direct child of the paragraph.

For a practical example, suppose you wish to embolden only those <li> elements that are direct children of <ol> elements. You can achieve this as follows, where the <li> elements that are direct children of <ul> elements do not get emboldened:

```
<!DOCTYPE html>
<html>
<head>
<style>
ol > li { font-weight:bold; }
</style>
</head>
<body>
<ol>
<li>One</li>
<li>Two</li>
<li>Three</li>
</ol>
<ul>
<li>One</li>
<li>Two</li>
<li>Three</li>
</ul>
</body>
</html>
```

The result of loading this HTML into a browser will be as follows:

```
1.One
2.Two
3.Three
•One
```

```
•Two
•Three
```

The ID Selector

If you give an element an ID name (like this: <div id='mydiv'>) then you can directly access it from CSS in the following way, which changes all the text in the element to italic:

```
#mydiv { font-style:italic; }
```

IDs can be used only once within a document, so only the first occurrence found will receive the new property value assigned by a CSS rule. But in CSS you can directly reference any IDs that have the same name, as long as they occur within different element types, like this:

```
<div id='myid'>Hello</div> <span
id='myid'>Hello</span>
```

Because IDs normally apply only to unique elements, the following rule will apply an underline to only the first occurrence of myid:

```
#myid { text-decoration:underline; }
```

However, you can ensure that CSS applies the rule to both occurrences like this:

```
span#myid { text-decoration:underline; } div#myid
{ text-decoration:underline; }
```

Or more succinctly like this (see “Selecting by Group” on page 435):

```
span#myid, div#myid { text-decoration:underline;
}
```

I don’t recommend using this form of selection because any JavaScript that also must access these elements cannot easily do so because the commonly used getElementByID() function will return only the first occurrence. To reference any other instances, a program would have to search through the whole list of elements in the document—a trickier task to undertake. So it’s generally better to always use unique ID names.

The Class Selector

When there are a number of elements in a page that you want to share the same styling, you can assign them all the same class name (like this: <span class=' myclass' >); then, create a single rule to modify all those elements at once, as in the following rule, which creates a 10-pixel left margin offset for all elements using the class:

```
.myclass { margin-left:10px; }
```

In modern browsers, you can have HTML elements use more than one class by separating the class names with spaces, like this: <span class='class1 class2 class3'>. Remember, though, that some very old browsers only allow a single class name in a class argument.

You can narrow the scope of action of a class by specifying the types of elements to which it should apply. For example, the following rule applies the setting only to paragraphs that use the class main:

```
p.main { text-indent:30px; }
```

In this example, only paragraphs using the class main (like this: <p class="main">) will receive the new property value. Any other element types that may try to use the class (such as <div class="main">) will not be affected by this rule.

The Attribute Selector

Many HTML tags support attributes, and using this type of selector can save you from having to use IDs and classes to refer to them. For example, you can directly reference attributes in the following manner, which sets all elements with the attribute type="sub mit" to a width of 100 pixels:

```
[type="submit"] { width:100px; }
```

If you wish to narrow down the scope of the selector to, for example, only form input elements with that attribute type, you could use the following rule instead:

```
form input[type="submit"] { width:100px; }
```

Attribute selectors also work on IDs and classes so that, for example, [ctass~="classname"] works exactly like the class selector .ctassname (except that the latter has a higher precedence). Likewise, [id="idname"] is equivalent to using the ID selector #idname. The class and ID selectors prefaced by # and . can therefore be viewed as shorthand for attribute selectors, but with a higher precedence. The ~= operator matches an attribute even if it is one of a space- separated group of attributes.

The Universal Selector

The * wildcard or universal selector matches any element, so the following rule will make a complete mess of a document by giving a green border to all of its elements:

```
* { border:1px solid green; }
```

It's therefore unlikely that you will use the * on its own, but as part of a compound rule it can be very powerful. For example, the following rule will apply the same styling as the preceding one, but only to all paragraphs that are sub-elements of the element with the ID boxout, and only as long as they are not direct children:

```
#boxout * p {border:1px solid green; }
```

Let's look at what's going on here. The first selector following #boxout is a * symbol, so it refers to any element within the boxout object. The following p selector then narrows down the selection focus by changing the selector to apply only to paragraphs (as defined by the p) that are sub-elements of elements returned by the * selector. Therefore, this CSS rule performs the following actions (in which I use the terms object and element interchangeably):

1. Find the object with the ID of boxout.
2. Find all sub-elements of the object returned in step 1.
3. Find all p sub-elements of the objects returned in step 2 and, because this is the final selector in the group, also find all p sub- and sub-sub-elements (and so on) of the objects returned in step 2.

4. Apply the styles within the {and } characters to the objects returned in step 3.

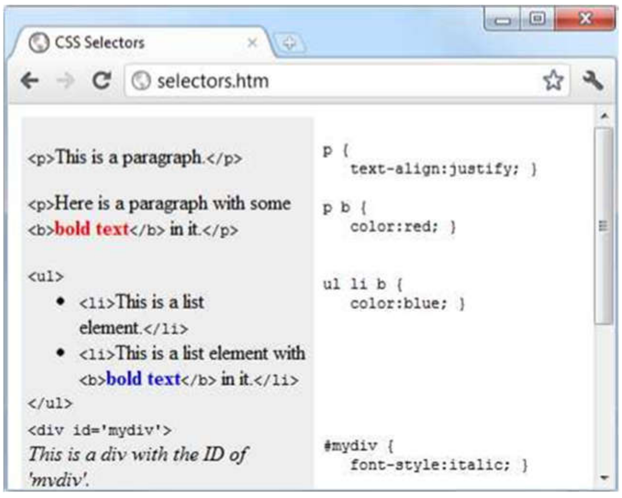
The net result of this is that the green border is applied only to paragraphs that are grandchildren (or great-grandchildren, etc.) of the main element.

Selecting by Group

Using CSS you can apply a rule to more than one element, class, or any other type of selector at the same time by separating the selectors with commas. So, for example, the following rule will place a dotted orange line underneath all paragraphs, the element with the ID of idname, and all elements that use the class classname:

```
p, #idname, .classname { border-bottom:1px dotted
orange; }
```

Figure 2-3 shows various selectors in use, with the rules applied to them alongside.



```
<p>This is a paragraph.</p>                  p {
                                                text-align:justify; }
<p>Here is a paragraph with some             p b {
<b>bold text</b> in it.</p>                     color:red; }

<ul>                                         ul li b {
  • <li>This is a list                          color:blue; }
    element.</li>
  • <li>This is a list element with
    <b>bold text</b> in it.</li>
</ul>
<div id='mydiv'>                             #mydiv {
This is a div with the ID of                    font-style:italic; }
'mydiv'.
```

Figure 2-3. Some HTML and the CSS rules used by it

Style Sheet Methods

Style sheets can be created via three different methods. In order of precedence from high to low, they are:

1. As inline styles
2. In an embedded style sheet
3. As an external style sheet

Again, these methods of style sheet creation are applied in reverse order of precedence. Therefore, all external style sheets are processed first, and their styles are applied to the document.

Next, any embedded styles (within <style> ... </style> tags) are processed, and any that conflict with external rules are given precedence and will override them.

Last, any styles applied directly to an element as an inline style (such as <div style="..."> ... </div>) are given the highest precedence, and override all previously assigned properties.

Style Sheet Selectors

There are three different ways of selecting elements to be styled. Going from highest to lowest order of precedence, they are:

- Referencing by individual ID or attribute selector
- Referencing in groups by class
- Referencing by element tags (such as <p> or <b>)

Selectors are processed according to the number and types of elements affected by a rule, which is a little different from the previous two methods for resolving conflicts. This is because rules do not have to apply only to one type of selector at a time, and may reference many different selectors.

Therefore, we need a method to determine the precedence of rules that can contain any combinations of selectors. It does this by calculating the specificity of each rule by ordering them from the widest to narrowest scope of action.

The Difference Between Div and Span Elements

Both <div> and <span> elements are types of containers, but with some different qualities. By default, a <div> element has infinite width (at least to the browser edge), which you can see by applying a border to one, like this:

```
<div styte="border:1px solid green;">Hello</div>
```

A <span> element, however, is only as wide as the text it contains. Therefore, the following line of HTML creates a border only around the word Hello, which does not extend to the righthand edge of the browser.

<span style="border:1px solid green;">Hello</span>

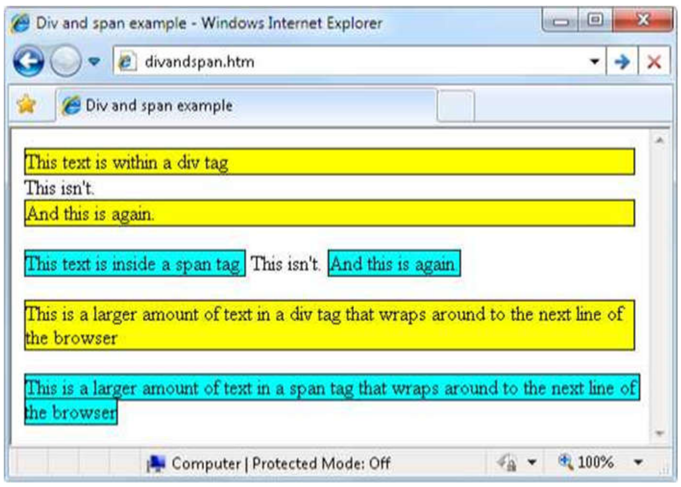
Also, <span> elements follow text or other objects as they wrap around, and can therefore have a complicated border. For example, in Example 19-2, I used CSS to make the background of all <div> elements yellow, to make all <span> elements cyan, and to add a border to both, before then creating a few example <span> and <div> sections.

Example 2-2. Div and span example

```
<!DOCTYPE html>
<html>
<head>
<title>Div and span example</title>
<style>
div, span { border :1px solid black; }
div { background-color:yellow;  }
span  { background-color:cyan; }
</style>
</head>
<body>
<div>This text is within a div tag</div>
This isn't. <div>And this is again.</div><br>
<span>This text is inside a span tag.</span>
This isn't. <span>And this is again.</span>
<br><br>
<div>This is a larger amount of text in a div that
wraps around to the next line of the browser</div><br>
<span>This is a larger amount of text in a span that
wraps around to the next line of the browser</span>
</body>
```

```
</html>
```

Figure 2-4 shows what this example looks like in a web browser. Although it is printed only in shades of gray in this book, the figure clearly shows how <div> elements extend to the righthand edge of a browser, and force the following content to appear at the start of the first available position below them.


Div and span example - Windows Internet Explorer
divandspan.htm
Div and span example
Computer | Protected Mode: Off
100%

This text is within a div tag
This isn't.
And this is again.

This text is inside a span tag This isn't. And this is again

This is a larger amount of text in a div tag that wraps around to the next line of the browser

This is a larger amount of text in a span tag that wraps around to the next line of the browser

Figure 2-4. A variety of elements of differing width

The figure also shows how <span> elements keep to themselves and take up only the space required to hold their content, without forcing subsequent content to appear below them.

For example, in the bottom two examples of the figure, you can also see that when <div> elements wrap around the screen edge they retain a rectangular shape, whereas <span> elements simply follow the flow of the text (or other contents) they contain.

Measurements

CSS supports an impressive range of units of measurement, enabling you to tailor your web pages precisely to specific values, or by relative dimensions. The ones I generally use (and believe you will also find the most useful) are pixels, points, ems, and percent, but here's the complete list:

Pixels

The size of a pixel varies according to the dimensions and pixel depth of the user's monitor. One pixel equals the width/height of a single dot on the screen, and so this measurement is best suited to monitors. For example:

```
.classname { margin:5px; }
```

Points

A point is equivalent in size to 1/72 of an inch. The measurement comes from a print design background and is best suited for that medium, but is also commonly used on monitors. For example:

```
.classname { font-size:14pt; }
```

Inches

An inch is the equivalent of 72 points and is also a measurement type best suited for print. For example:

```
.classname { width:3in; }
```

Centimeters

Centimeters are another unit of measurement best suited for print. One centimeter is a little over 28 points. For example:

```
.classname { height:2cm; }
```

Millimeters

A millimeter is 1/10 of a centimeter (or almost 3 points). Millimeters are another measure best suited to print. For example:

```
.classname { font-size:5mm; }
```

Picas

A pica is another print typographic measurement, which is equivalent to 12 points. For example:

```
.classname { font-size:1pc; }
```

Ems

An em is equal to the current font size and is therefore one of the more useful measurements for CSS because it is used to describe relative dimensions. For example:

```
.classname { font-size:2em; }
```

Exs

An ex is also related to the current font size; it is equivalent to the height of a lowercase letter x. This is a less popular unit of measurement that is most often used as a good approximation for helping to set the width of a box that will contain some text. For example:

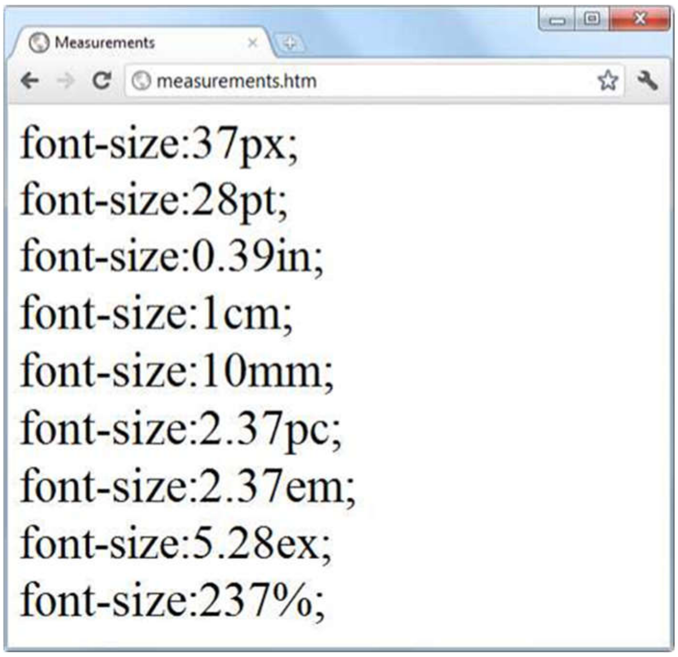
```
.classname { width:20ex; }
```

Percent

This unit is related to the em in that it is exactly 100 times greater (when used on a font). Whereas 1 em equals the current font size, the same size is 100 in percent. When not relating to a font, this unit is relative to the size of the container of the property being accessed. For example:

```
.classname { height:120%; }
```

Figure 2-5 shows each of these measurement types in turn being used to display text in almost identical sizes.



```
font-size:37px;
font-size:28pt;
font-size:0.39in;
font-size:1cm;
font-size:10mm;
font-size:2.37pc;
font-size:2.37em;
font-size:5.28ex;
font-size:237%;
```

Figure 2-5. Different measurements that display almost the same

Fonts and Typography

There are four main font properties that you can style using CSS: family, style, size, and weight. Between them, you can fine-tune the way text displays in your web pages and/or when printed.

font-family

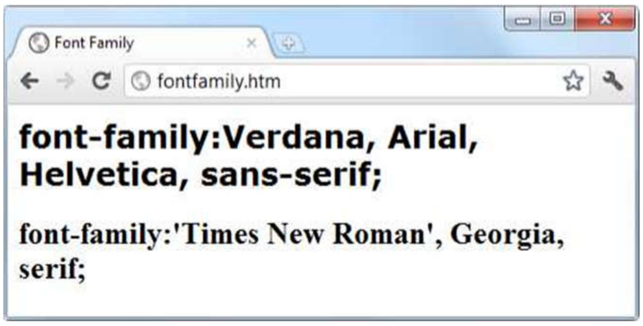
The font-family property assigns the font to use. It also supports listing a variety of fonts in order of preference from left to right, so that styling can fall back gracefully when the user doesn't have the preferred font installed. For example, to set the default font for paragraphs, you might use a CSS rule such as this:

```
    p { font-family:Verdana, Arial, Helvetica,
sans-serif; }
```

Where a font name is made up of two or more words, you must enclose the name in quotation marks, like this:

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

Figure 2-6 shows these two sets of CSS rules being applied.



font-family:Verdana, Arial, Helvetica, sans-serif;

font-family:'Times New Roman', Georgia, serif;

Figure 2-6. Selecting font families

font-style

With the font-style property you can choose to display a font normally, in italics, or obliquely. The following rules create three classes (normal, italic, and oblique) that can be applied to elements to create these effects:

```
    .normal { font-style:normal; }
    .italic { font-style:italic; }
    .oblique { font-style:oblique; }
```

font-size

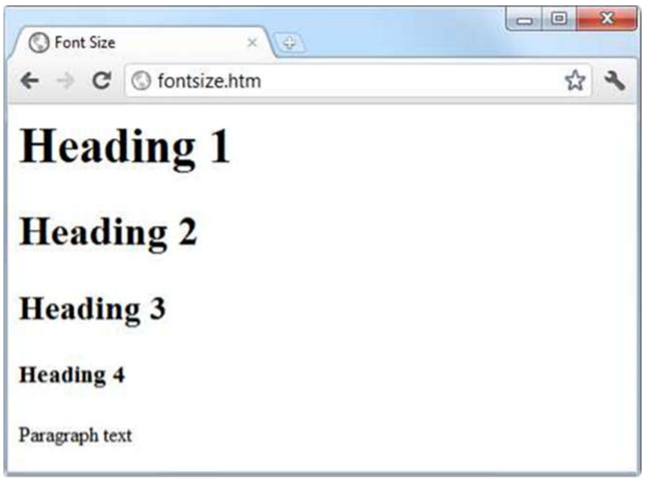
As described in the earlier section on measurements, there are a large number of ways you can change a font's size. But these all boil down to two main types: fixed and relative. A fixed setting looks like the following rule, which sets the default paragraph font size to 14 point:

```
    p { font-size:14pt; }
```

Alternatively, you may wish to work with the current default font size, using it to style various types of text such as headings. In the following rules, relative sizes of some headers are defined, with the <h4> tag starting off 20% bigger than the default, and with each greater size another 40% larger than the previous one:

```
H1 { font-size:240%; }
H2 { font-size:200%; }
H3 { font-size:160%; }
H4 { font-size:120%; }
```

Figure 2-7 shows a selection of font sizes in use.



Heading 1

Heading 2

Heading 3

Heading 4

Paragraph text

Figure 2-7. Setting four heading sizes and the default paragraph size

font-weight

Using the font-weight property you can choose how boldly to display a font. It supports a number of values, but the main ones you will use are likely to be normal and bold, like this:

```
.bold { font-weight:bold; }
```

CSS Colors

You can apply colors to the foreground and background of text and objects using the color and background-color properties (or by supplying a single argument to the background property). The colors specified can be one of the named colors (such as red or blue), colors created from hexadecimal RGB triplets (such as #ff0000 or #0000ff), or colors created using the rgb CSS function.

The standard 16 color names as defined by the W3C standards organization are: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. The following rule uses one of these names to set the background color for an object with the ID of object:

```
#object { background-color:silver; }
```

In this rule, the foreground color of text in all <div> elements is set to yellow (because on a computer display, hexadecimal levels of ff red, plus ff green, plus 00 blue creates the color yellow):

```
div { color:#ffff00; }
```

Or, if you don't wish to work in hexadecimal, you can specify your color triplets using the rgb function, as in the following rule, which changes the background color of the current document to aqua:

```
body { background-color:rgb(0, 255, 255); }
```

Short Color Strings

There is also a short form of the hex digit string in which only the first of each 2-byte pair is used for each color. For example, instead of assigning the color #fe4692, you instead use #f49, omitting the second hex digit from each pair, which equates to a color value of #ff4499.

This results in almost the same color and is useful where exact colors are not required. The difference between a six-digit and three-digit string is that the former supports 16 million different colors, while the latter supports four thousand.

Wherever you intend to use a color such as #883366, this is the direct equivalent of #836 (because the repeated digits are implied by the shorter version), and you can use either string to create the exact same color.

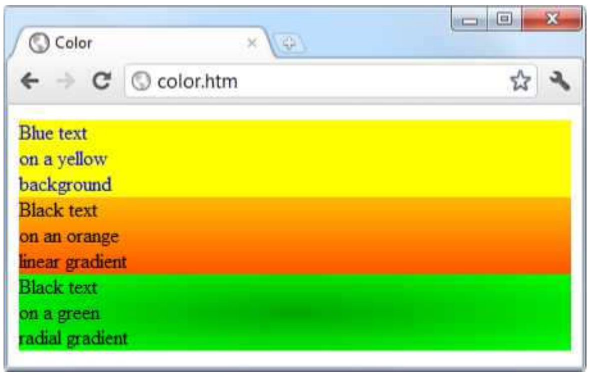
Gradients

In place of using a solid background color, you can choose to apply a gradient, which will then automatically flow from a given initial color to a final color of your choice. It is best used in conjunction with a simple color rule so that browsers that don't support gradients will at least display a solid color.

Example 2-3 uses a rule to display an orange gradient (or simply plain orange on nonsupporting browsers) as shown in the middle section of Figure 2-10.

Example 2-3. Creating a linear gradient

```
<!DOCTYPE html>
<html>
<head>
<title>Creating a linear gradient</title>
<style>
.orangegrad { background:orange;
background:linear-gradient(top, #fb0, #f50);
background:-moz-linear-gradient(top, #fb0, #f50);
background:-webkit-linear-gradient(top, #fb0, #f50);
background:-o-linear-gradient(top, #fb0, #f50);
background:-ms-linear-gradient(top, #fb0, #f50); }
</style>
</head>
<body>
<div class='orangegrad'>Black text<br> on an
orange<br>linear gradient</div>
</body>
</html>
```



Blue text
on a yellow
background
Black text
on an orange
linear gradient
Black text
on a green
radial gradient

Figure 2-10. A solid background color, a linear gradient, and a radial gradient

To create a gradient, choose where it will begin out of top, bottom, left, right, and center (or any combination, such as top left or center right), enter the start and end colors you require, and then apply either the linear-gradient or radial- gradient rule, making sure you also supply rules for all browsers that you are targeting.

You can also use more than just a start and end color by also supplying what are termed stop colors in between as additional arguments. In this case, for example, if five arguments are supplied, each argument will control the color change over a fifth of the area represented by its location in the argument list.

Positioning Elements

Elements within a web page fall where they are placed in the document, but you can move them about by changing an element's position property from the default of static to one of absolute, relative, or fixed.

Absolute Positioning

An element with absolute positioning is removed from the document, and any other elements that are capable will flow into its released space. You can then position the object anywhere you like within the document using the top, right, bottom, and left properties. It will then rest on top of (or behind) other elements.

So, for example, to move an object with the ID of object to the absolute location of 100 pixels down from the document start and 200 pixels in from the left, you would apply the following rules to it (you can also use any of the other units of measurement supported by CSS):

```
#object {
position:absolute; top :100px;
left :200px;
}
```

Relative Positioning

Likewise, you can move the object relative to the location it would occupy in the normal document flow. So, for example, to move object 10 pixels down and 10 pixels to the right of its normal location, you would use the following rules:

```
#object {
position:relative; top :10px;
left  :10px;
}
```

Fixed Positioning

The final positioning property setting lets you move an object to an absolute location, but only within the current browser viewport. Then, when the document is scrolled, the object remains exactly where it has been placed, with the main document scrolling beneath it—a great way to create dock bars and other similar devices. To fix the object to the top-left corner of the browser window, you would use the following rules:

```
#object {
position:fixed; top  :0px;
left  :0px;
}
```

In Figure 2-11, Example 2-4 has been loaded into a browser, and the browser has been reduced in width and height so that you must scroll down to see all of the web page.

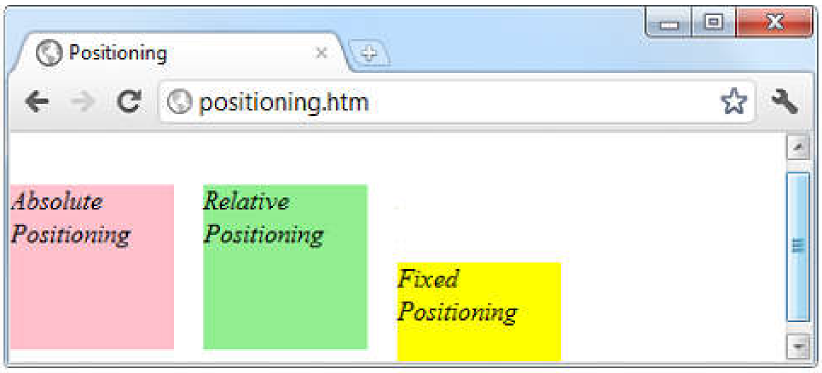


Figure 2-11. Using different positioning values

When this is done, it is immediately obvious that the element with fixed positioning remains in place even through scrolling. You can also see that the element with absolute positioning is located exactly at 100 pixels down, with 0 horizontal offset, while the element with relative positioning is actually moved up by 8 pixels and then offset from the left margin by 110 pixels in order to line up alongside the first element.

Example 2-4. Applying different positioning values

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

```
<head>
<title>Positioning</title>
<style>
#object1 {
position :absolute;
background:pink;
width :100px;
height :100px;
top :100px;
left :0px;
}
#object2 {
position :relative;
background:lightgreen;
width :100px;
height :100px;
top :-8px;
left :110px;
}
#object3 {
position :fixed;
background:yellow;
width :100px;
height :100px;
top :100px;
left :236px;
}
</style>
</head>
<body>
<br><br><br><br><br>
```

```
<div id='object1'>Absolute Positioning</div>
<div id='object2'>Relative Positioning</div>
<div id='object3'>Fixed Positioning</div>
</body>
</html>
```

In the figure, the element with fixed positioning initially lines up with the other two elements, but has stayed put while the others have been scrolled up the page, and now appears offset below them.

The Box Model and Layout

The CSS properties affecting the layout of a page are based around the box model (see Chapter 14 for more details), a nested set of properties surrounding an element. Virtually all elements have (or can have) these properties, including the document body, whose margin you can, for example, remove with the following rule:

```
body { margin:0px; }
```

The box model of an object starts at the outside, with the object’s margin. Inside this is the border, then there is padding between the border and the inner contents, and finally there’s the object’s contents.

Once you have the hang of the box model, you will be well on your way to creating professionally laid-out pages, as these properties alone will make up much of your page styling.

Setting Margins

The margin is the outermost level of the box model. It separates elements from each other and its use is quite smart. For example, assume you have chosen to give a number of elements a default margin of 10 pixels around each. When they are placed on top of each other, this would create a gap of 20 pixels (the total of the adjacent border widths).

CSS overcomes this potential issue, however: when two elements with borders are positioned directly one above the other, only the larger of the two margins is used to separate them. If both margins are the same width, just one of the widths is used. This way, you are much more likely to get the result you want. But you should note that the margins of absolutely positioned or inline elements do not collapse.

The margins of an element can be changed en masse with the margin property, or individually with margin-left, margin-top, margin-right, and margin-bottom. When setting the margin property, you can supply one, two, three, or four arguments, which have the effects commented in the following rules:

```
/* Set all margins to 1 pixel */ margin:1px;
/* Set top and bottom to 1 pixel, and left and right to 2 */ margin:1px 2px;
/* Set top to 1 pixel, left and right to 2, and bottom to 3 */ margin:1px 2px 3px;
/* Set top to 1 pixel, right to 2, bottom to 3, and left to 4 */ margin:1px 2px 3px 4px;
```

Figure 2-13 shows Example 2-6 loaded into a browser, with the margin property rule (highlighted in bold) applied to a square element that has been placed inside a table element. The table has been given no dimensions, so it will simply wrap as closely around the inner <div> element as it can. As a consequence, there is a margin of 10 pixels above it, 20 pixels to its right, 30 pixels below it, and 40 pixels to its left.

Example 2-6. How margins are applied

```
<!DOCTYPE html>
<html>
<head>
<title>Margins</title>
<style>
#object1 {
background :lightgreen;
border-style:solid;
border-width:1px;
font-family :"Courier New";
font-size :9px;
width :100px;
height :100px;
padding :5px;
margin :10px 20px 30px 40px;
}
table {
padding :0;
border :1px solid black;
background :cyan;
}
</style>
</head>
<body>
<table>
<tr>
<td>
<div id='object1'>margin:<br>10px 20px 30px 40px;</div>
</td>
</tr>
</table>
</body>
</html>
```

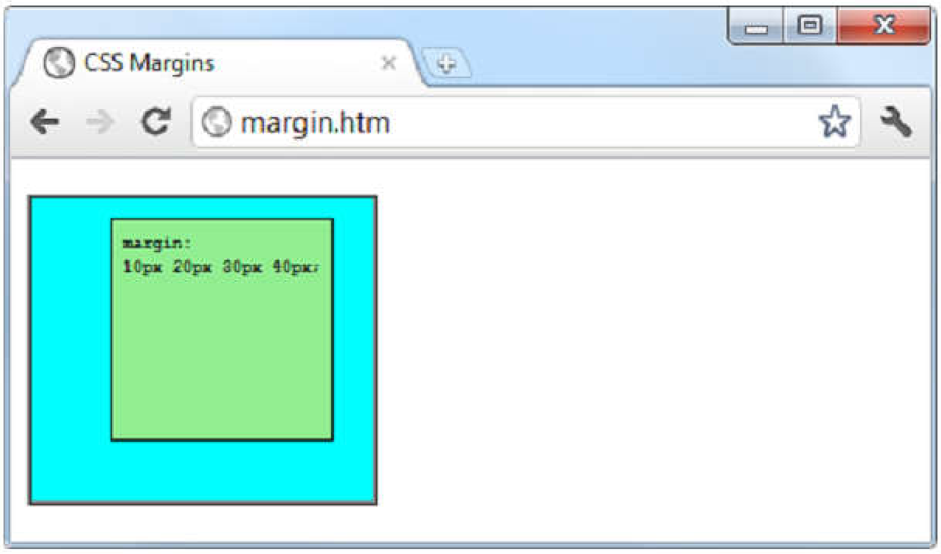


Figure 2-13. The outer table expands according to the margin widths

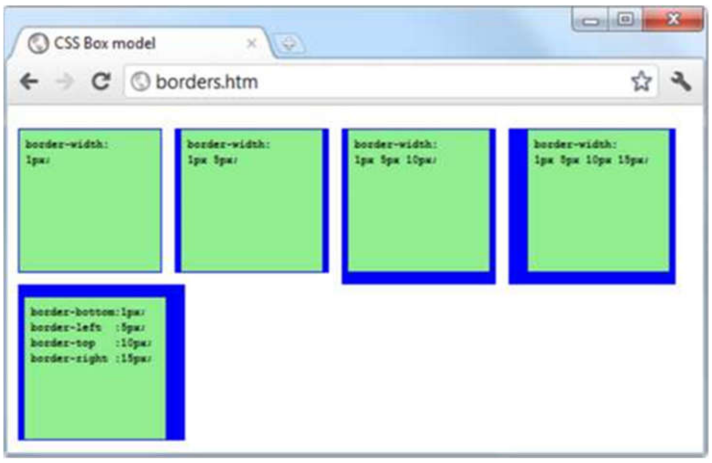
Applying Borders

The border level of the box model is similar to the margin except that there is no collapsing. It is the next level as we move into the box model. The main properties used to modify borders are border, border-left, border-top, border-right, and border- bottom, and each ofthese can have other subproperties added as suffixes, such as - color, -style,and -width.

The four ways to access individual property settings used for the margin property also apply with the border-width property, so all the following are valid rules:

```
/* All borders */ border-width:1px;
/* Top/bottom left/right */ border-width:1px 5px;
/* Top left/right bottom */ border-width:1px 5px 10px;
/* Top right bottom left */ border-width:1px 5px 10px 15px;
```

Figure 2-14 shows each of these rules applied in turn to a group of square elements. In the first one, you can clearly see that all borders have a width of 1 pixel. The second element, however, has a top and bottom border width of 1 pixel, while its side widths are 5 pixels each.



```
border-width:
1px;
```

```
border-width:
1px 5px;
```

```
border-width:
1px 5px 10px;
```

```
border-width:
1px 5px 10px 15px;
```

```
border-bottom:1px;
border-left  :5px;
border-top   :10px;
border-right :15px;
```

Figure 2-14. Applying long- and shorthand border rule values

The third element has a 1 pixel wide top, its sides are 5 pixels wide, and its bottom is 10 pixels wide. The fourth element has a 1-pixel top border width, a 5-pixel right border width, a 10-pixel bottom border width, and a 15-pixel left border width.

The final element, under the previous ones, doesn't use the shorthand rules; instead, it has each of the border widths set separately. As you can see, it takes a lot more typing to achieve the same result.

Adjusting Padding

The deepest of the box model levels (other than the contents of an element) is the padding, which is applied inside any borders and/or margins. The main properties used to modify padding are padding, padding-left, padding-top, padding-right, and padding-bottom.

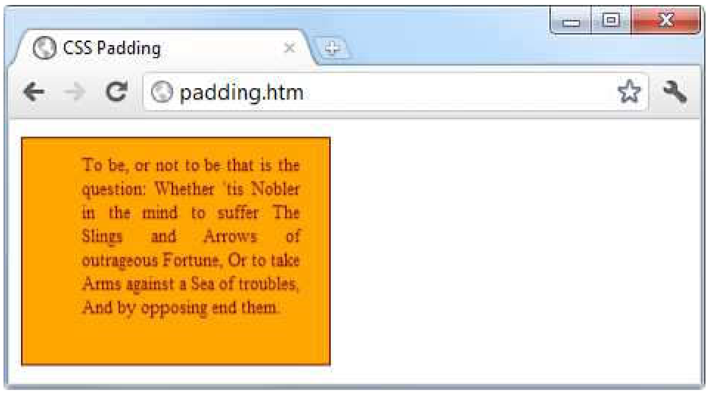
The four ways of accessing individual property settings used for the margin and border properties also apply with the padding property, so all the following are valid rules:

```
/* All padding */ padding:1px;
/* Top/bottom and left/right */ padding:1px 2px;
/* Top, left/right and bottom */ padding:1px 2px 3px;
/* Top, right, bottom and left */ padding:1px 2px 3px 4px;
```

Figure 2-15 shows the padding rule (shown in bold) in Example 2-7 applied to some text within a table cell (as defined by the rule display:table-cell;, which makes the encapsulating <div> element display like a table cell), which has been given no dimensions so it will simply wrap as closely around the text as it can. As a consequence there is padding of 10 pixels above the inner element, 20 pixels to its right, 30 pixels below it, and 40 pixels to its left.

Example 2-7. Applying padding

```
<!DOCTYPE html>
<html>
<head>
<title>Padding</title>
<style>
#object1 {
border-style:solid;   border-width:1px;   background
:orange; color :darkred;
font-family :Arial;
font-size  :12px;
text-align :justify;
display  :table-cell;
width :148px;
padding  :10px 20px   30px 40px; }
</style>
</head>
<body>
<div id='object1'>To be, or not to be that is the
question: Whether 'tis Nobler in the mind to suffer The
Slings and Arrows of outrageous Fortune,
Or to take Arms against a Sea of troubles,
And by opposing end them.</div>
</body>
</html>
```



To be, or not to be that is the question: Whether 'tis Nobler in the mind to suffer The Slings and Arrows of outrageous Fortune, Or to take Arms against a Sea of troubles, And by opposing end them.

Figure 2-15. Applying different padding values to an object

Object Contents

Deep within the box model levels, at its center, lies an element that can be styled in all the ways discussed in this chapter, and which can (and usually will) contain further subelements, which in turn may contain sub-sub-elements, and so on, each with its own styling and box model settings.

CSS3 Borders

CSS3 also brings a lot more flexibility to the way borders can be presented, by allowing you to independently change the colors of all four border edges, to display images for the edges and corners, to provide a radius value for applying rounded corners to borders, and to place box shadows underneath elements.

The border-color Property

There are two ways you can apply colors to a border. First, you can pass a single color to the property, as follows:

```
border-color:#888;
```

This property sets all the borders of an element to mid-gray. You can also set border colors individually, like this (which sets the border colors to various shades of gray):

```
border-top-color :#000; border-left-color :#444;
border-right-color :#888; border-bottom-color:#ccc;
```

You can also set all the colors individually with a single declaration, as follows: border-color:#f00 #0f0 #880 #00f;

This declaration sets the top border color to #f00, the right one to #0f0, the bottom one to #880, and the left one to #00f (red, green, orange, and blue, respectively). You can also use color names for the arguments.

The border-radius Property

Prior to CSS3, talented web developers came up with numerous different tweaks and fixes in order to achieve rounded borders, generally using <table> or <div> tags.

But now adding rounded borders to an element is really simple, and it works on the latest versions of all major browsers, as shown in Figure 20-3, in which a 10-pixel border is displayed in different ways. Example 20-2 shows the HTML for this.

Example 2-8. The border-radius property <!DOCTYPE html>

```
<html> <!-- borderradius.html -->
<head>
<title>CSS3 Border Radius Examples</title>
<style>
.box {
margin-bottom:10px;
font-family :'Courier New', monospace; font-size :12pt;
text-align :center;

padding :10px;
width   :380px;
height  :75px;
border  :10px solid #006;
}
.b1 {
-moz-border-radius :40px; -webkit-border-radius:40px;
border-radius  :40px;
}
.b2 {
```

```
-moz-border-radius :40px 40px 20px 20px;
-webkit-border-radius:40px 40px 20px 20px; border-radius
  :40px 40px 20px 20px;
}
.b3 {
-moz-border-radius-topleft  :20px;
-moz-border-radius-topright :40px;
-moz-border-radius-bottomleft :60px;
-moz-border-radius-bottomright   :80px;
-webkit-border-top-left-radius   :20px;
-webkit-border-top-right-radius  :40px;
-webkit-border-bottom-left-radius  :60px;
-webkit-border-bottom-right-radius:80px;
border-top-left-radius :20px;
border-top-right-radius  :40px;
border-bottom-left-radius   :60px;
border-bottom-right-radius  :80px;
}
.b4 {
-moz-border-radius-topleft
-moz-border-radius-topright
-moz-border-radius-bottomleft
-moz-border-radius-bottomright
-webkit-border-top-left-radius
-webkit-border-top-right-radius
-webkit-border-bottom-left-radius
-webkit-border-bottom-right-radius
border-top-left-radius
border-top-right-radius
border-bottom-left-radius
border-bottom-right-radius
```

```
border-radius:40px;
</div> <div class='box b2'>
border-radius:40px 40px 20px 20px;
<div class='box b3'>
border-top-left-radius &nbsp;&nbsp;&nbsp;:20px;<br>
border-top-right-radius &nbsp;&nbsp;:40px;<br>
border-bottom-left-radius :60px;<br>
border-bottom-right-radius:80px;
</div>
<div class='box b4'>
border-top-left-radius &nbsp;&nbsp;&nbsp;:40px 20px;<br>
border-top-right-radius &nbsp;&nbsp;:40px 20px;<br>
border-bottom-left-radius :20px 40px;<br>
border-bottom-right-radius:20px 40px;
</div>
</body>
</html>
```

So, for example, to create a rounded border with a radius of 20 pixels, you could simply use the following declaration:

```
border-radius:20px;
```

Although most browsers will work fine with border radius properties (including IE), some current (and many older) versions of the major browsers use different property names. So, if you wish to support them all, you will need to also use the relevant browser-specific prefixes, such as -moz- and -webktt-. To ensure that Example 20-2 works in all browsers, I have included all the required prefixes.

You can specify a separate radius for each of the four corners, like this (applied in a clockwise direction starting from the top-left corner):

```
border-radtus:10px 20px 30px 40px;
```

If you prefer, you can also address each corner of an element individually, like this:

```
border-top-left-radius :20px;
border-top-right-radius :40px; border-bottom-left-radius
:60px; border-bottom-right-radius:80px;
```

And, when referencing individual corners, you can supply two arguments to choose a different vertical and horizontal radius (giving more interesting and subtle borders) like this:

```
border-top-left-radius :40px 20px;
border-top-right-radius :40px 20px;
border-bottom-left-radius :20px 40px;
border-bottom-right-radius:20px 40px;
```

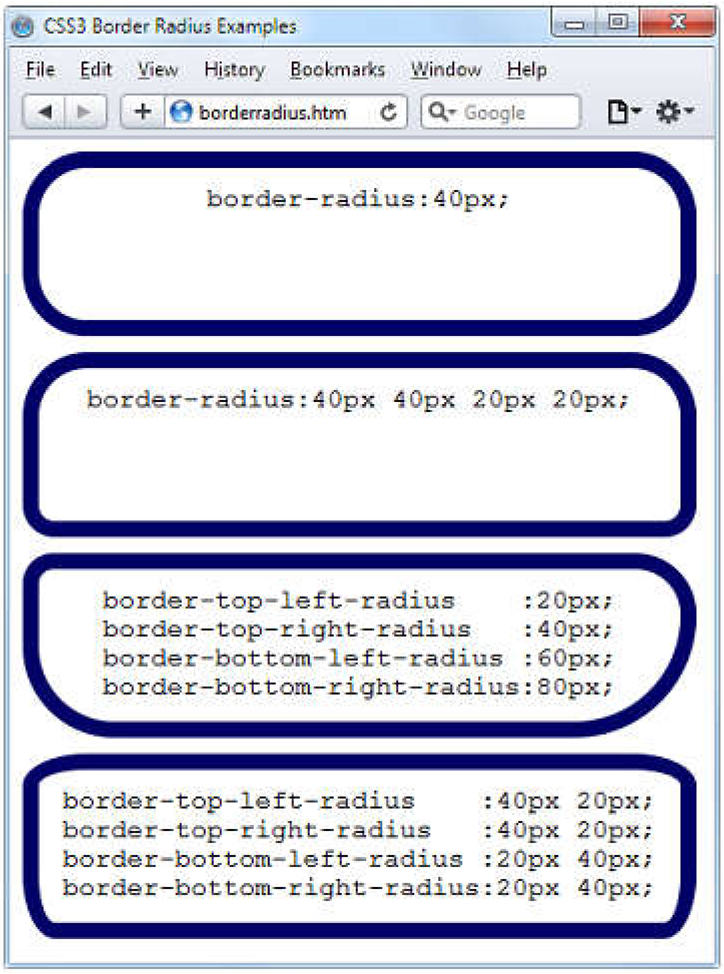
The first argument is the horizontal, and the second is the vertical radius.

Box Shadows

To apply a box shadow, specify a horizontal and vertical offset from the object, the amount of blurring to add to the shadow, and the color to use, like this:

box-shadow:15px 15px 10px #888;

The two instances of 15px specify the vertical and horizontal offset from the element, and these values can be negative, zero, or positive. The 10px specifies the amount of blurring, with smaller values resulting in less blurring. And the #888 is the color for the shadow, which can be any valid color value. The result of this declaration can be seen in Figure 2-16.


CSS3 Border Radius Examples
File Edit View History Bookmarks Window Help
borderradius.htm
Google

```
border-radius:40px;
```

```
border-radius:40px 40px 20px 20px;
```

```
border-top-left-radius    :20px;
border-top-right-radius   :40px;
border-bottom-left-radius :60px;
border-bottom-right-radius:80px;
```

```
border-top-left-radius    :40px 20px;
border-top-right-radius   :40px 20px;
border-bottom-left-radius :20px 40px;
border-bottom-right-radius:20px 40px;
```

Figure 2-16. Mixing and matching various border radius properties

2.5. Discussion Questions

1. Which directive do you use to import one style sheet into another (or the <style> section of some HTML)?
2. What HTML tag can you use to import a style sheet into a document?
3. Which HTML tag attribute is used to directly embed a style into an element?
4. What is the difference between a CSS ID and a CSS class?
5. Which characters are used to prefix (a) IDs, and (b) class names in a CSS rule?
6. In CSS rules, what is the purpose of the semicolon?
7. How can you add a comment to a style sheet?
8. Which character is used by CSS to represent any element?
9. How can you select a group of different elements and/or element types in CSS?
10. Given a pair of CSS rules with equal precedence, how can you make one have

greater precedence over the other?

11. How do you set up a transition on an object so that when any of its properties are changed, the change will transition immediately in a linear fashion over the course of half a second?

2.6. Topics for self-study

1. Denwer and Dreamweaver Applications
2. Working with tables in HTML
3. Working with hyperlinks and forms in HTML
4. Cookies in JavaScript
5. Creating objects in JavaScript
6. String functions in PHP
7. Creating pagination in PHP
8. Providing security in PHP and MySQL
9. Using AJAX technology in developing dynamic websites
10. Using Content Management Systems (Joomla, Wordpress)

2.7. Creating multimedia course with CourseLab

To support students' self-education we have created interactive multimedia course with the help of CourseLab software. This course contains lecture texts with examples, and there is also CSS test.

The course creation process is very simple and user-friendly and very similar to creating presentations in PowerPoint. Main screen three main parts: Course structure, Slide thumbnails and Main editing area (Figure 2-16).

For every module it is possible to create a title page and a master page. The master page will contain elements that will be displayed one every slide. The title page will contain information that will be displayed on starting the module. To switch between the three sections, use the template buttons (Figure 2-17).

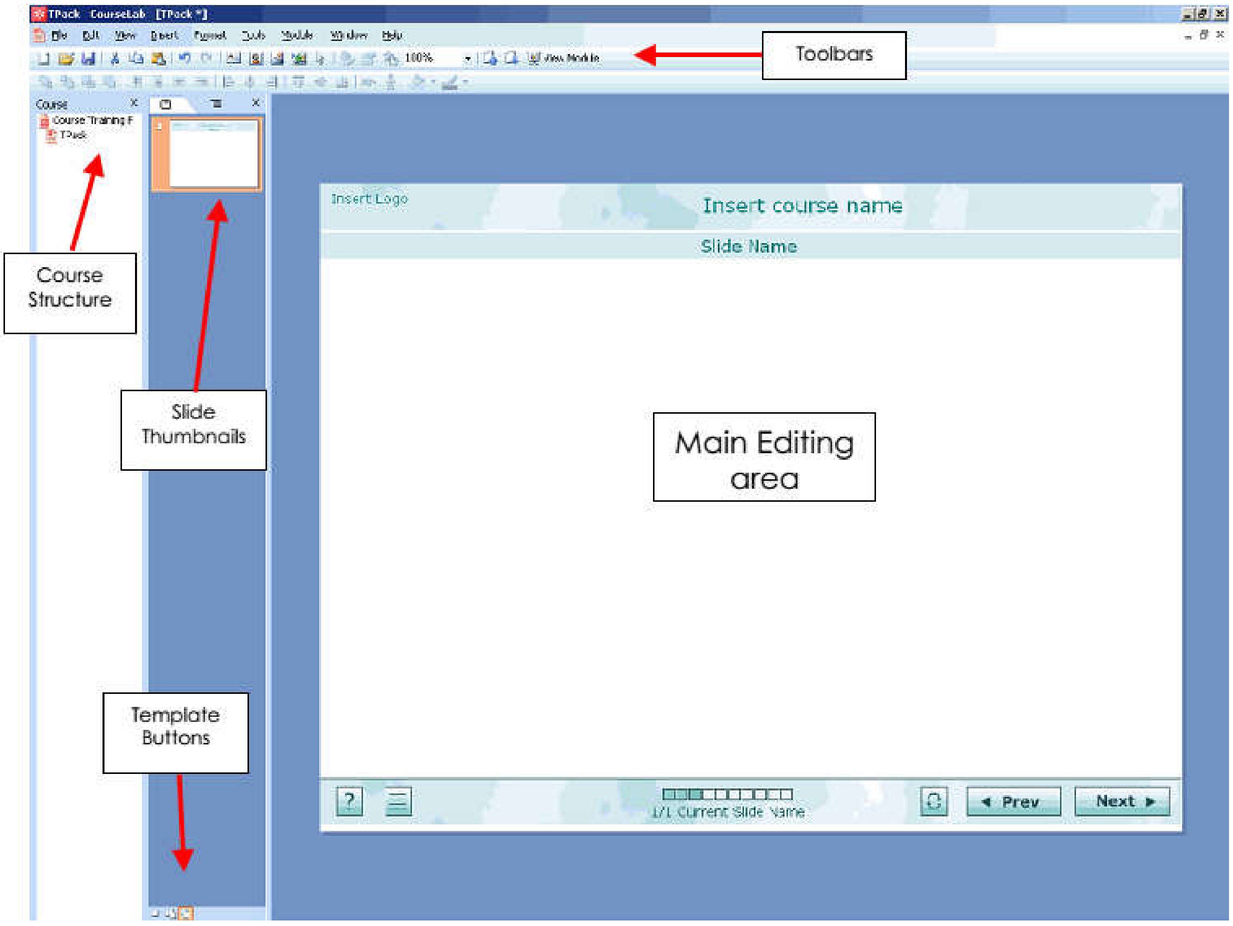


Figure 2-16. CourseLab main screen

For every module it is possible to create a title page and a master page. The master page will contain elements that will be displayed one every slide. The title page will contain information that will be displayed on starting the module. To switch between the three sections, use the template buttons

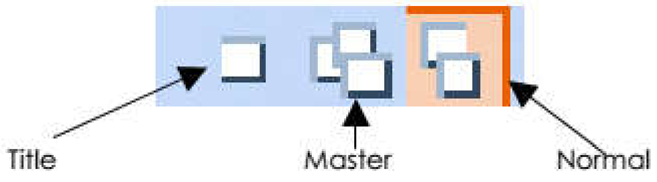


Figure 2-17. Template buttons

Slides and Frames

Each slide can contain content made up of text, graphics and/or CourseLab objects. In addition a slide can be broken down into frames that allow step by step processes to be displayed (Figure 2-18).

An example frame structure can be seen below. Most modules that are created will make use of a slide by slide basis, but frames can be used when trying to simulate software.

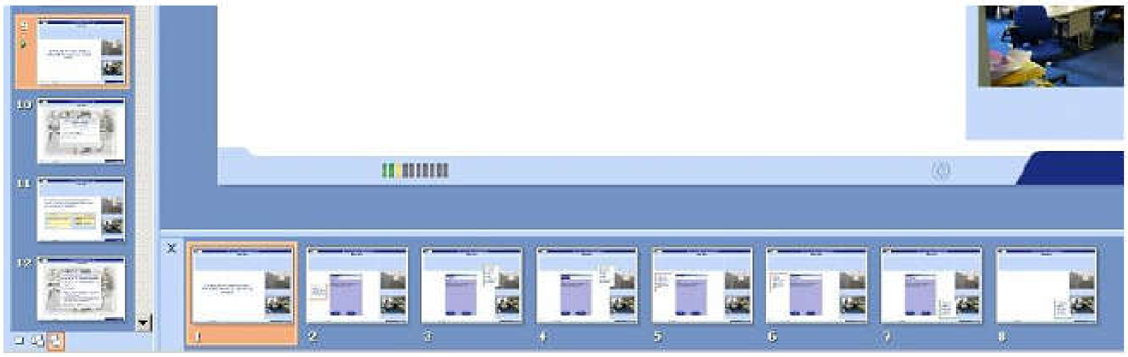


Figure 2-18. Slides and Frames

Adding text

To insert text onto a slide, click the text box on the toolbar. This will display a text box on the screen. Double click this box to open the text editor. The text editor has functions similar to Microsoft Word. However it does not have any language tools such as Spell Check or Thesaurus. In order to make sure that your text is correct, copy and paste from the Editor to Word, spell check and the copy back again.

2.8. Using Case Study

Analysing a case study requires you to practice applying your knowledge and your thinking skills to a real situation. To learn from a case study analysis you will be analysing, applying knowledge, reasoning and drawing conclusions. Using the Case study in Web-programming course may be the most important part of the education process, because it is maximally close to the real life processes and it helps to improve practical web-development skills.

Below we give an example of standart case that can be given to students.

Example Case:

In this case study we build the home page of a website from scratch that has layout, navigation bars, side bars and everything else needed to make a decent looking website.

Case Study Lessons

In the Case study we begin by looking at a website proposal and set up the basic environment needed for the future lessons. After this we go through the proposal isolating the comments and presentation required for each layer of the site. We will start by creating a layout for the site and then fill in the various parts of the layout in the following lessons.

Lesson 1: Website Proposal

In this lesson we read a website proposal from a potential client whom we shall call 'Fine Fancy Foods' for want of a better name. This will give us a general outline of what the customer wants from the site and it's our job to decipher the proposal and turn it into a website. After reading through the proposal we will set our environment up for use by later lessons.

Lesson 2: Page Layout

In lesson 6 of the CSS Advanced Tutorials we looked at dividing a webpage into sections using the <div> tag. In this lesson we expand on that knowledge to create a layout for our homepage. We will highlight the appropriate sections of the proposal, that will give us an idea of the layout and create the relevant CSS from this information.

Lesson 3: Website Banner

In this lesson we create a banner for our homepage. We will highlight the appropriate section of the proposal and look at the Fine Fancy Foods Layout Diagram. This will give us the information required to create the banner and the CSS and HTML we need to achieve it.

Lesson 4: Navigation Bar

One of the most important areas of website design is allowing users easy navigation to the other content on our site. In this lesson we create a navigation bar for our homepage for this purpose. Using the relevant section of the proposal and the Fine Fancy Foods Layout Diagram we can see what the client wants with regards to this part of the site. With this information acquired we can create the navigation bar, using CSS and HTML.

Lesson 5: Main Content

In this lesson we create the main content for our homepage. We will highlight the appropriate section of the proposal and look at the Fine Fancy Foods Layout Diagram. Along with the files sent to us by the client we will have the information required to create the main content part of our informational data.

Lesson 6: Left Sidebar

Ok with the main content HTML and CSS out of the way it is time to focus on the other parts of the page that make up the informational content. These areas being the left and right sidebars and the footer. In this lesson we create the left sidebar for our homepage. We will highlight the appropriate section of the proposal and look at the Fine Fancy Foods Layout Diagram once again. With this information and the appropriate files sent to us by the client we will have everything needed to create the left sidebar.

Lesson 7: Right Sidebar

Having completed the left sidebar we can now move on to the right sidebar content for our homepage. We will highlight the appropriate section of the proposal and look at the Fine Fancy Foods Layout Diagram to see the layout. With this information and the relevant files sent to us by the client we will have everything required to create the right sidebar.

Lesson 8: Footer Bar

The left and right sidebars are done so now all that is left to complete the content for our homepage is the footer bar. After highlighting the appropriate sections of the proposal and looking at the Fine Fancy Foods Layout Diagram again, we will have the information to create the footer bar and finish the client site.

Lesson 9: Case Study Summary

In this summary we list the various CSS and HTML used to construct the homepage for Fine Fancy Foods. We start by showing the file structure used for this client, which could be applied to any site creation. This summary completes the lessons for the site.

SUMMARY

The presence of educational technology is growing in the classroom. The new generation of kids come ready to work with these new technologies, which play an important role in children's learning and acquiring various cognitive knowledge so that educational technology must be incorporated into future curricula. The application of educational technology enhances skills and cognitive characteristics. With the help of new technology comes an explosion of learning and receiving new information, especially on mobile devices.

Teachers have been using new technologies in the classroom. However, the development and application of new technologies grows as a measure that is the question of whether teachers are trained to keep up with them. Here we have two problems. Are the teachers have the ability to use educational technology and whether the school is sufficiently equipped with all modern technical means? Numerous studies were carried out, some are still ongoing, but we have to find the right strategies to apply educational technology in teaching.

In the Chapter 1 we provided information about nowadays state of Web technologies and described some issues of teaching the Web-development course in universities, and also described such educational technologies as brainstorming, case study and clustering. In the Chapter 2 we provided lessons plan and technological maps, lecture texts, assesments, cases, CourseLab multimedia presentations and other materials related to the Web development course. In this thesis, we described the importance of using case study and brainstorming in developing self-working abilities of students, and described the full process of teaching the CSS basics to the students. Based on these knowledges, students can learn the whole subject which is not covered in the lectures on their own, and that helps the universities produce more appropriate specialists.

We have also analysed the effectiveness of using assessments, brainstorming and clustering educational technologies while teaching cascading style sheets. We have also learned the process of creation of high-quality interactive multimedia presentations with the help of CourseLab tool.

Currently the evolution of the Web is one of the most crucial and demanding challenges in the computer science. Teaching students the practice of creating high-quality web applications and websites is the key to successful development of the computer science industry in our country.

I would like to give a recommendation to our teacher in the field of Web science to provide a wide variety of modern web technologies in their courses and take into account new directions and technologies, and try to get rid of out-of-date information.

References

1. The resolution of the President of the Republic of Uzbekistan from March 26, 2013 of № PP-1942 “About measures for further enhancement of system of training in the field of information and communication technologies”
2. Robin Nixon – Learning PHP, MySQL, JavaScript, CSS & HTML5, 3rd Edition - 2014.
3. DUAN Jie, SUN Xiangyang, CAO Weiguo, XUE Kejuan – Methods to Improve the Teaching Effect of the Professional Course of the University
4. Ji, G.. Theory of distance education. The Central Radio and TV University Press.
5. Clark, R. E.. Reconsidering the research on learning from media
6. Dr. Lazar Stošić, Aleksinac Serbia – The importance of educational technology in teaching
7. Ronald V. Morris, Kathryn M.Obenchain – Three Methods for Teaching the Social Studies to Students through the Arts
8. http://writing2.richmond.edu/

APPENDIX

Case Study

In this case study we build the home page of a website from scratch that has layout, navigation bars, side bars and everything else needed to make a decent looking website.

Case Study Lessons

In the Case study we begin by looking at a website proposal and set up the basic environment needed for the future lessons. After this we go through the proposal isolating the comments and presentation required for each layer of the site. We will start by creating a layout for the site and then fill in the various parts of the layout in the following lessons.

Lesson 1: Website Proposal

In this lesson we read a website proposal from a potential client whom we shall call 'Fine Fancy Foods' for want of a better name. This will give us a general outline of what the customer wants from the site and it's our job to decipher the proposal and turn it into a website. After reading through the proposal we will set our environment up for use by later lessons.

Lesson 2: Page Layout

In lesson 6 of the CSS Advanced Tutorials we looked at dividing a webpage into sections using the <div> tag. In this lesson we expand on that knowledge to create a layout for our homepage. We will highlight the appropriate sections of the proposal, that will give us an idea of the layout and create the relevant CSS from this information.

Lesson 3: Website Banner

In this lesson we create a banner for our homepage. We will highlight the appropriate section of the proposal and look at the Fine Fancy Foods Layout Diagram. This will give us the information required to create the banner and the CSS and HTML we need to achieve it.

Lesson 4: Navigation Bar

One of the most important areas of website design is allowing users easy navigation to the other content on our site. In this lesson we create a navigation bar for our homepage for this purpose. Using the relevant section of the proposal and the Fine Fancy Foods Layout Diagram we can see what the client wants with regards to this part of the site. With this information acquired we can create the navigation bar, using CSS and HTML.

Lesson 5: Main Content

In this lesson we create the main content for our homepage. We will highlight the appropriate section of the proposal and look at the Fine Fancy Foods Layout Diagram. Along with the files sent to us by the client we will have the information required to create the main content part of our informational data.

Lesson 6: Left Sidebar

Ok with the main content HTML and CSS out of the way it is time to focus on the other parts of the page that make up the informational content. These areas being the left and right sidebars and the footer. In this lesson we create the left sidebar for our homepage. We will highlight the appropriate section of the proposal and look at the Fine Fancy Foods Layout Diagram once again. With this information and the appropriate files sent to us by the client we will have everything needed to create the left sidebar.

Lesson 7: Right Sidebar

Having completed the left sidebar we can now move on to the right sidebar content for our homepage. We will highlight the appropriate section of the proposal and look at the Fine Fancy Foods Layout Diagram to see the layout. With this information and the relevant files sent to us by the client we will have everything required to create the right sidebar.

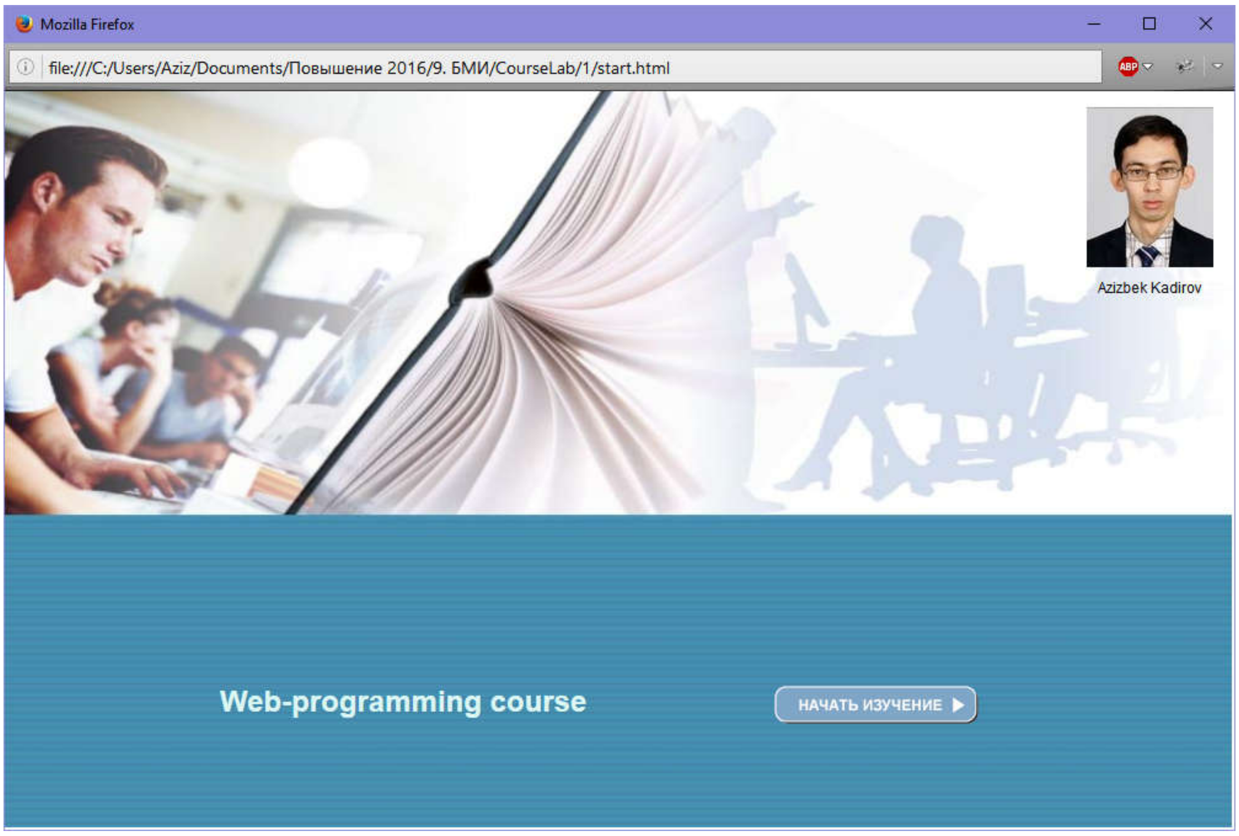
Lesson 8: Footer Bar

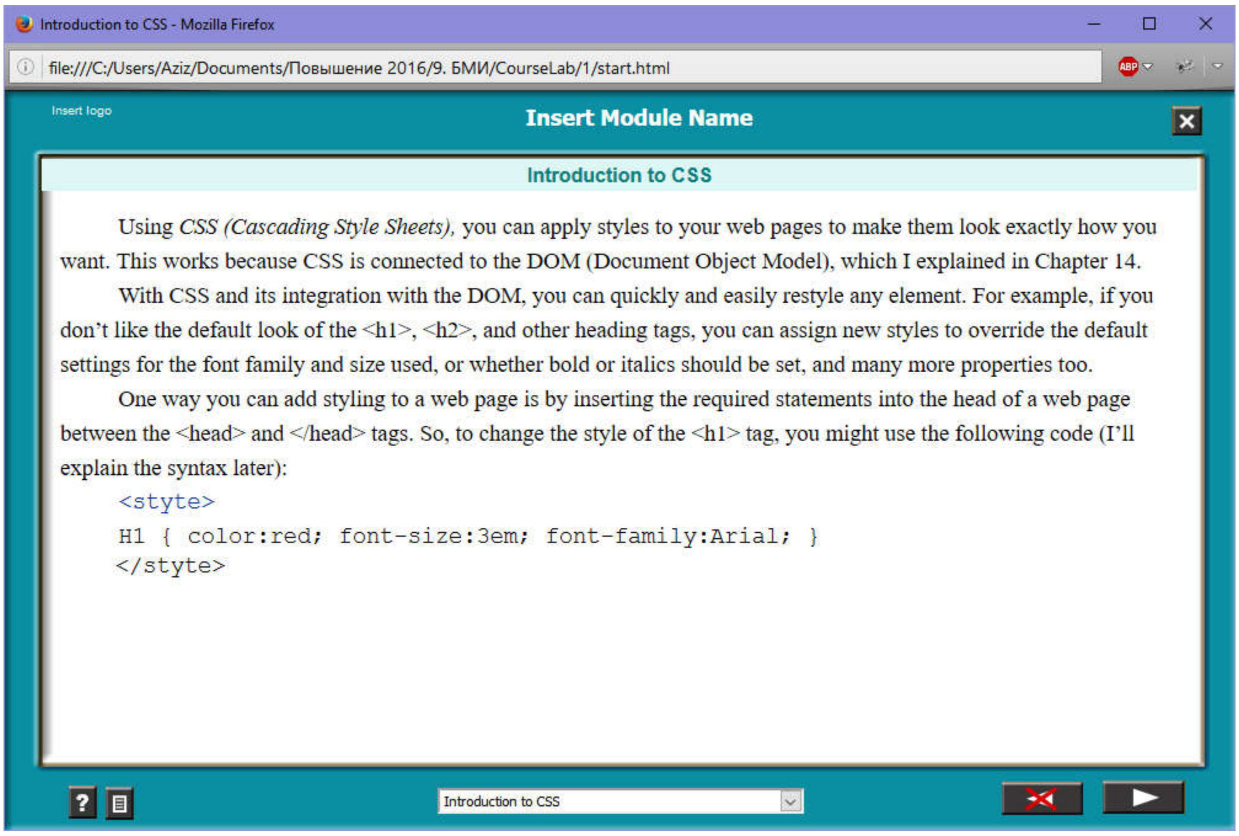
The left and right sidebars are done so now all that is left to complete the content for our homepage is the footer bar. After highlighting the appropriate sections of the proposal and looking at the Fine Fancy Foods Layout Diagram again, we will have the information to create the footer bar and finish the client site.

Lesson 9: Case Study Summary

In this summary we list the various CSS and HTML used to construct the homepage for Fine Fancy Foods. We start by showing the file structure used for this client, which could be applied to any site creation. This summary completes the lessons for the site.

CourseLab presentation


Mozilla Firefox
file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html
Azizbek Kadirov
Web-programming course
НАЧАТЬ ИЗУЧЕНИЕ

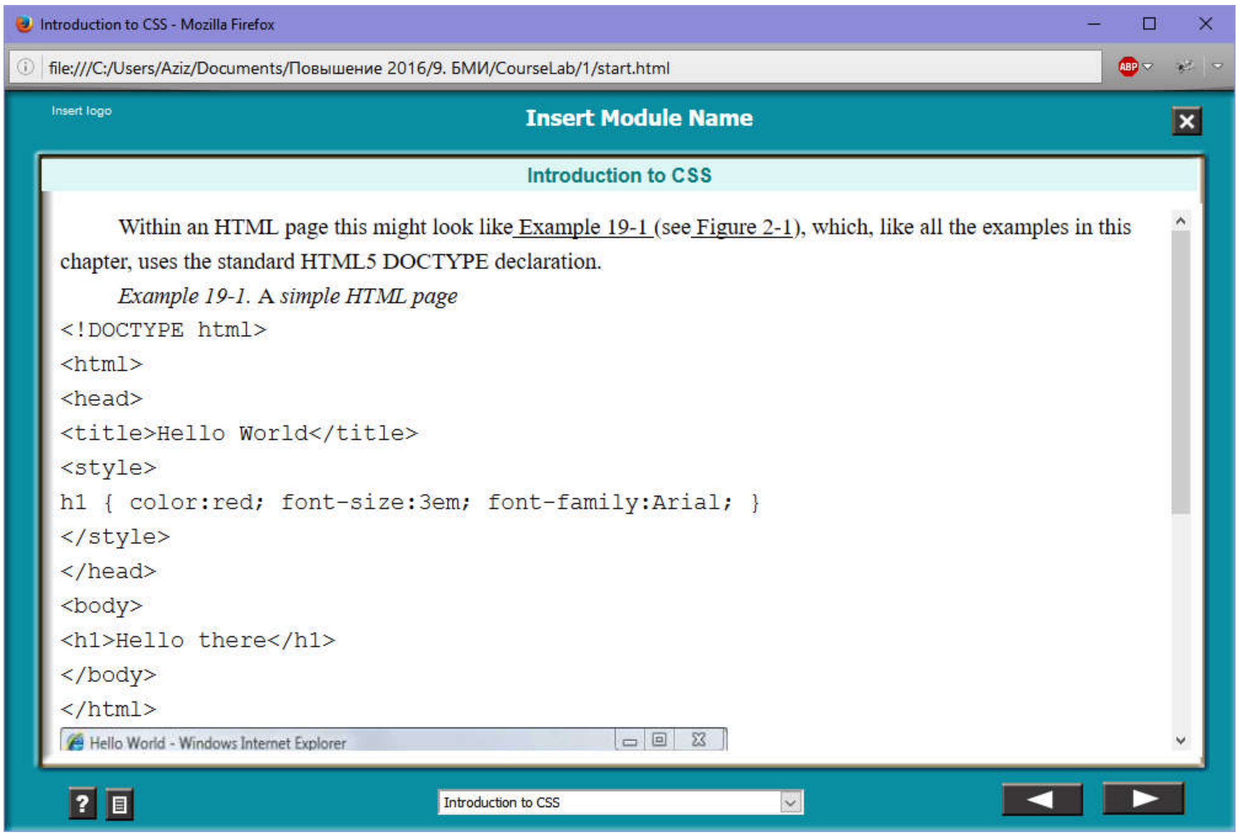

Introduction to CSS - Mozilla Firefox
file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html
Insert logo
Insert Module Name
Introduction to CSS

Using *CSS (Cascading Style Sheets),* you can apply styles to your web pages to make them look exactly how you want. This works because CSS is connected to the DOM (Document Object Model), which I explained in Chapter 14.

With CSS and its integration with the DOM, you can quickly and easily restyle any element. For example, if you don't like the default look of the <h1>, <h2>, and other heading tags, you can assign new styles to override the default settings for the font family and size used, or whether bold or italics should be set, and many more properties too.

One way you can add styling to a web page is by inserting the required statements into the head of a web page between the <head> and </head> tags. So, to change the style of the <h1> tag, you might use the following code (I'll explain the syntax later):

```
<styte>
H1 { color:red; font-size:3em; font-family:Arial; }
</styte>
```



Introduction to CSS - Mozilla Firefox

file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html

Insert logo

Insert Module Name

Introduction to CSS

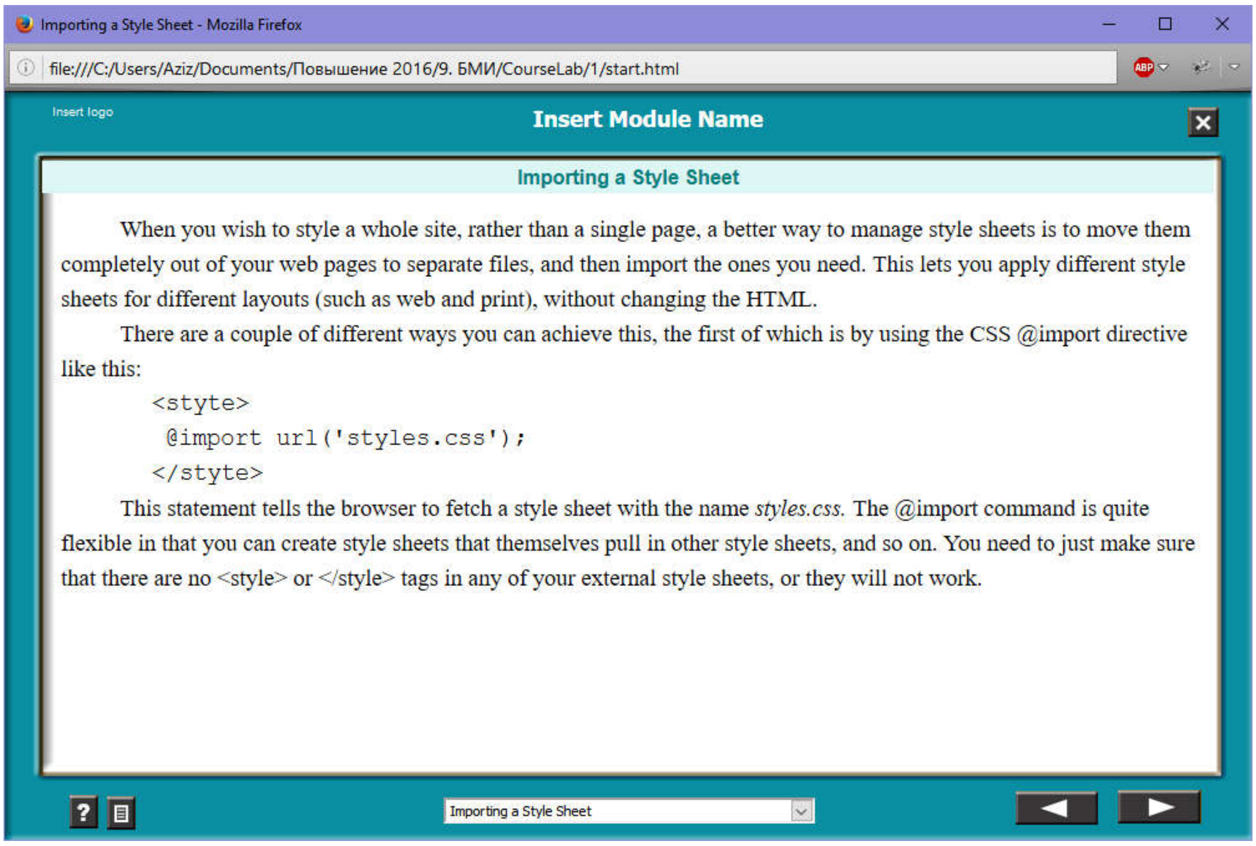
Within an HTML page this might look like Example 19-1 (see Figure 2-1), which, like all the examples in this chapter, uses the standard HTML5 DOCTYPE declaration.

Example 19-1. A simple HTML page

```
<!DOCTYPE html>
<html>
<head>
<title>Hello World</title>
<style>
h1 { color:red; font-size:3em; font-family:Arial; }
</style>
</head>
<body>
<h1>Hello there</h1>
</body>
</html>
```

Hello World - Windows Internet Explorer

Introduction to CSS



Importing a Style Sheet - Mozilla Firefox

file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html

Insert logo

Insert Module Name

Importing a Style Sheet

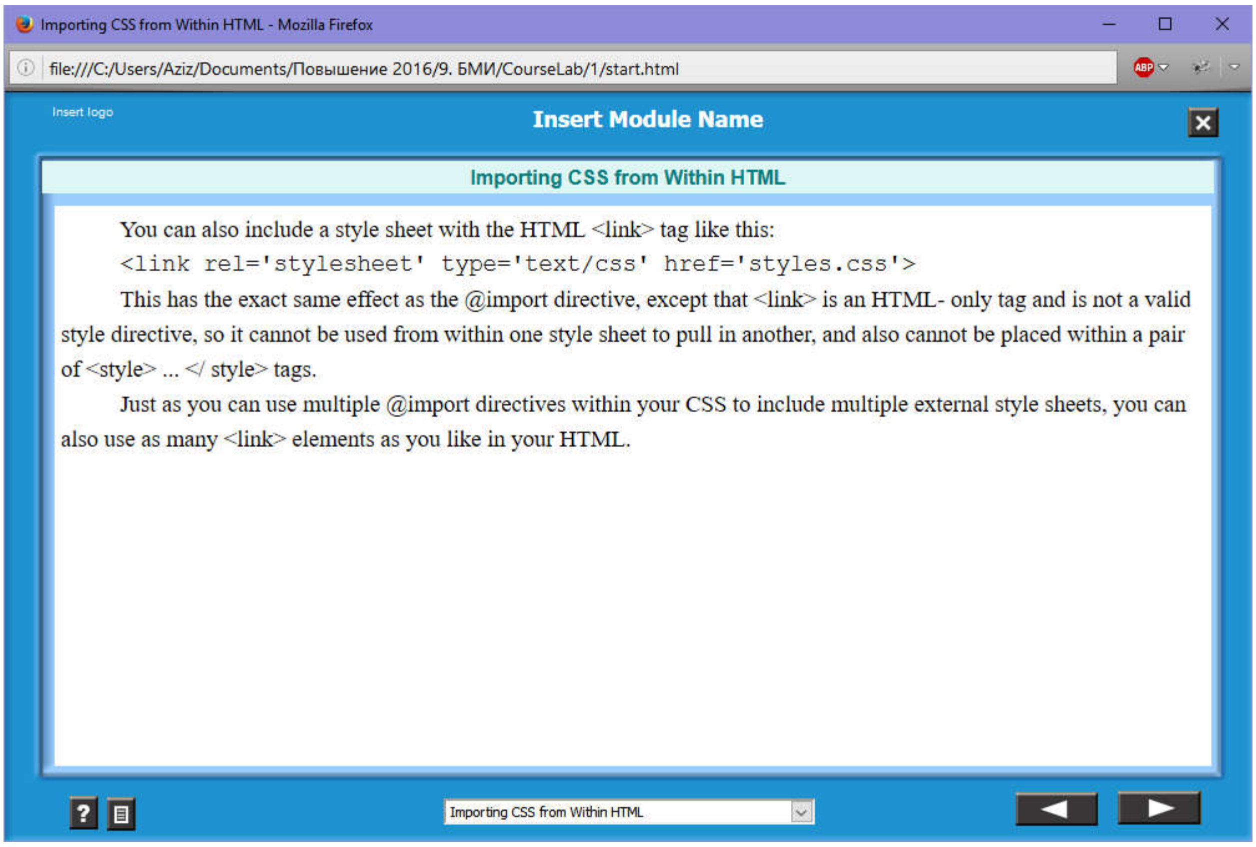
When you wish to style a whole site, rather than a single page, a better way to manage style sheets is to move them completely out of your web pages to separate files, and then import the ones you need. This lets you apply different style sheets for different layouts (such as web and print), without changing the HTML.

There are a couple of different ways you can achieve this, the first of which is by using the CSS @import directive like this:

```
<styte>
 @import url('styles.css');
</styte>
```

This statement tells the browser to fetch a style sheet with the name *styles.css*. The @import command is quite flexible in that you can create style sheets that themselves pull in other style sheets, and so on. You need to just make sure that there are no <style> or </style> tags in any of your external style sheets, or they will not work.

Importing a Style Sheet


Importing CSS from Within HTML - Mozilla Firefox

file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html

Insert logo

Insert Module Name

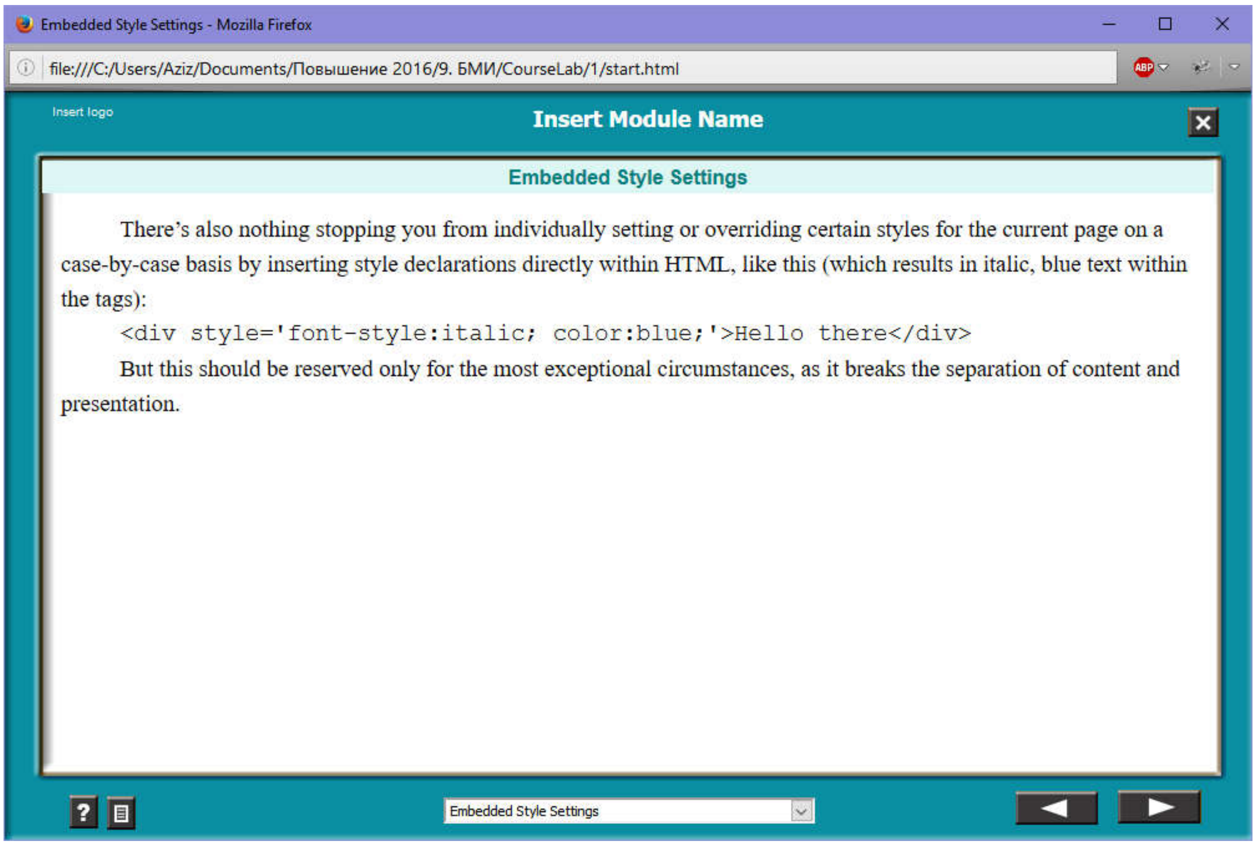
Importing CSS from Within HTML

You can also include a style sheet with the HTML <link> tag like this:

```
<link rel='stylesheet' type='text/css' href='styles.css'>
```

This has the exact same effect as the @import directive, except that <link> is an HTML- only tag and is not a valid style directive, so it cannot be used from within one style sheet to pull in another, and also cannot be placed within a pair of <style> ... </ style> tags.

Just as you can use multiple @import directives within your CSS to include multiple external style sheets, you can also use as many <link> elements as you like in your HTML.


Embedded Style Settings - Mozilla Firefox

file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html

Insert logo

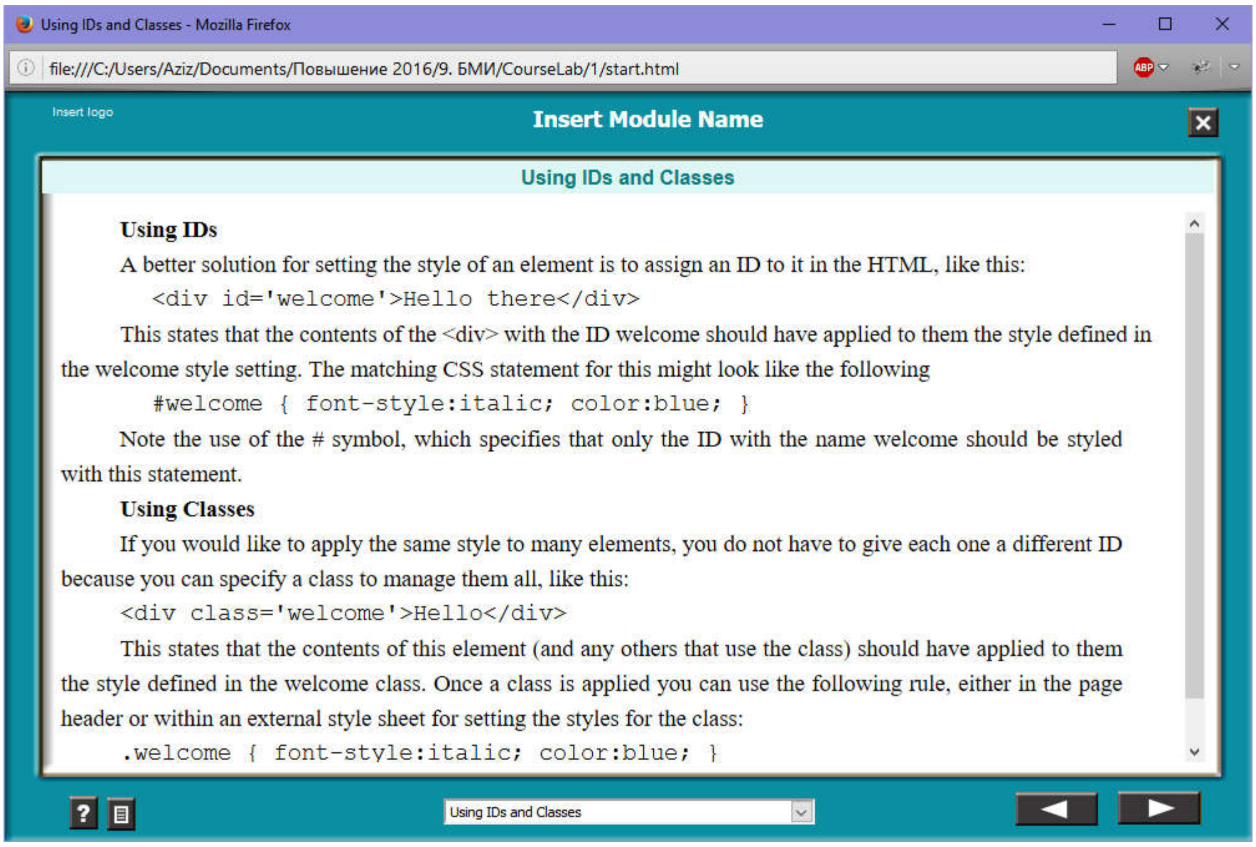
Insert Module Name

Embedded Style Settings

There’s also nothing stopping you from individually setting or overriding certain styles for the current page on a case-by-case basis by inserting style declarations directly within HTML, like this (which results in italic, blue text within the tags):

```
<div style='font-style:italic; color:blue;'>Hello there</div>
```

But this should be reserved only for the most exceptional circumstances, as it breaks the separation of content and presentation.


Using IDs and Classes - Mozilla Firefox
file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html
Insert logo
Insert Module Name
Using IDs and Classes

Using IDs

A better solution for setting the style of an element is to assign an ID to it in the HTML, like this:

```
<div id='welcome'>Hello there</div>
```

This states that the contents of the <div> with the ID welcome should have applied to them the style defined in the welcome style setting. The matching CSS statement for this might look like the following

```
#welcome { font-style:italic; color:blue; }
```

Note the use of the # symbol, which specifies that only the ID with the name welcome should be styled with this statement.

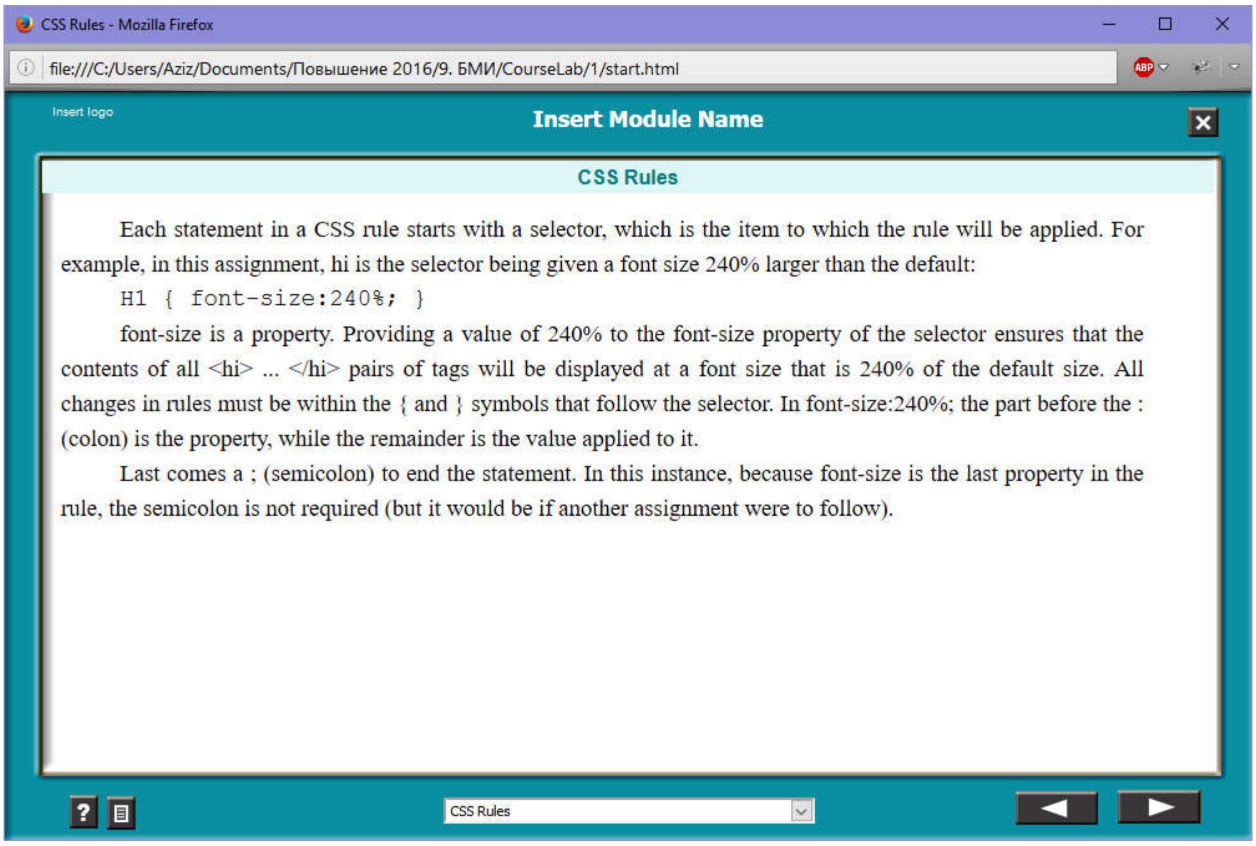
Using Classes

If you would like to apply the same style to many elements, you do not have to give each one a different ID because you can specify a class to manage them all, like this:

```
<div class='welcome'>Hello</div>
```

This states that the contents of this element (and any others that use the class) should have applied to them the style defined in the welcome class. Once a class is applied you can use the following rule, either in the page header or within an external style sheet for setting the styles for the class:

```
.welcome { font-style:italic; color:blue; }
```

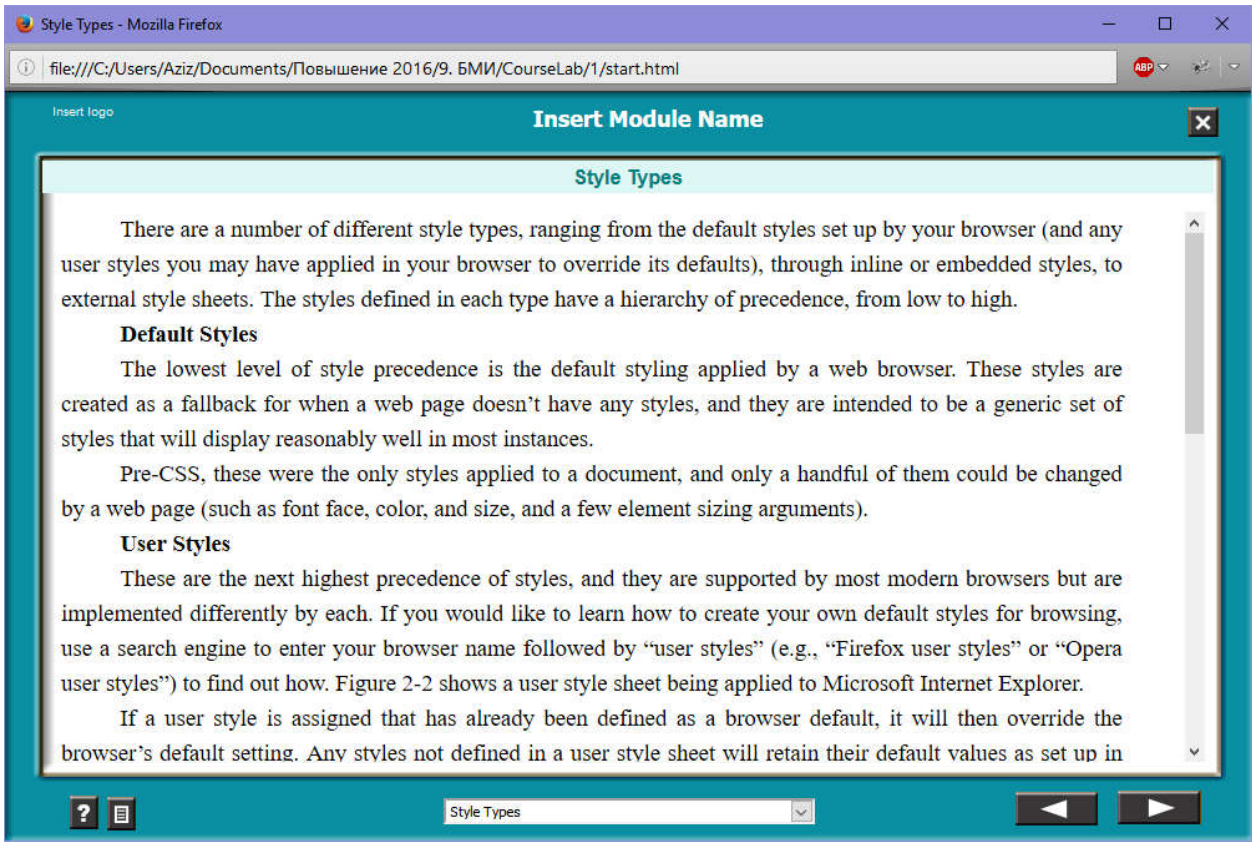

CSS Rules - Mozilla Firefox
file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html
Insert logo
Insert Module Name
CSS Rules

Each statement in a CSS rule starts with a selector, which is the item to which the rule will be applied. For example, in this assignment, hi is the selector being given a font size 240% larger than the default:

```
H1 { font-size:240%; }
```

font-size is a property. Providing a value of 240% to the font-size property of the selector ensures that the contents of all <hi> ... </hi> pairs of tags will be displayed at a font size that is 240% of the default size. All changes in rules must be within the { and } symbols that follow the selector. In font-size:240%; the part before the : (colon) is the property, while the remainder is the value applied to it.

Last comes a ; (semicolon) to end the statement. In this instance, because font-size is the last property in the rule, the semicolon is not required (but it would be if another assignment were to follow).


Style Types - Mozilla Firefox
file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html
Insert logo
Insert Module Name

Style Types

There are a number of different style types, ranging from the default styles set up by your browser (and any user styles you may have applied in your browser to override its defaults), through inline or embedded styles, to external style sheets. The styles defined in each type have a hierarchy of precedence, from low to high.

Default Styles

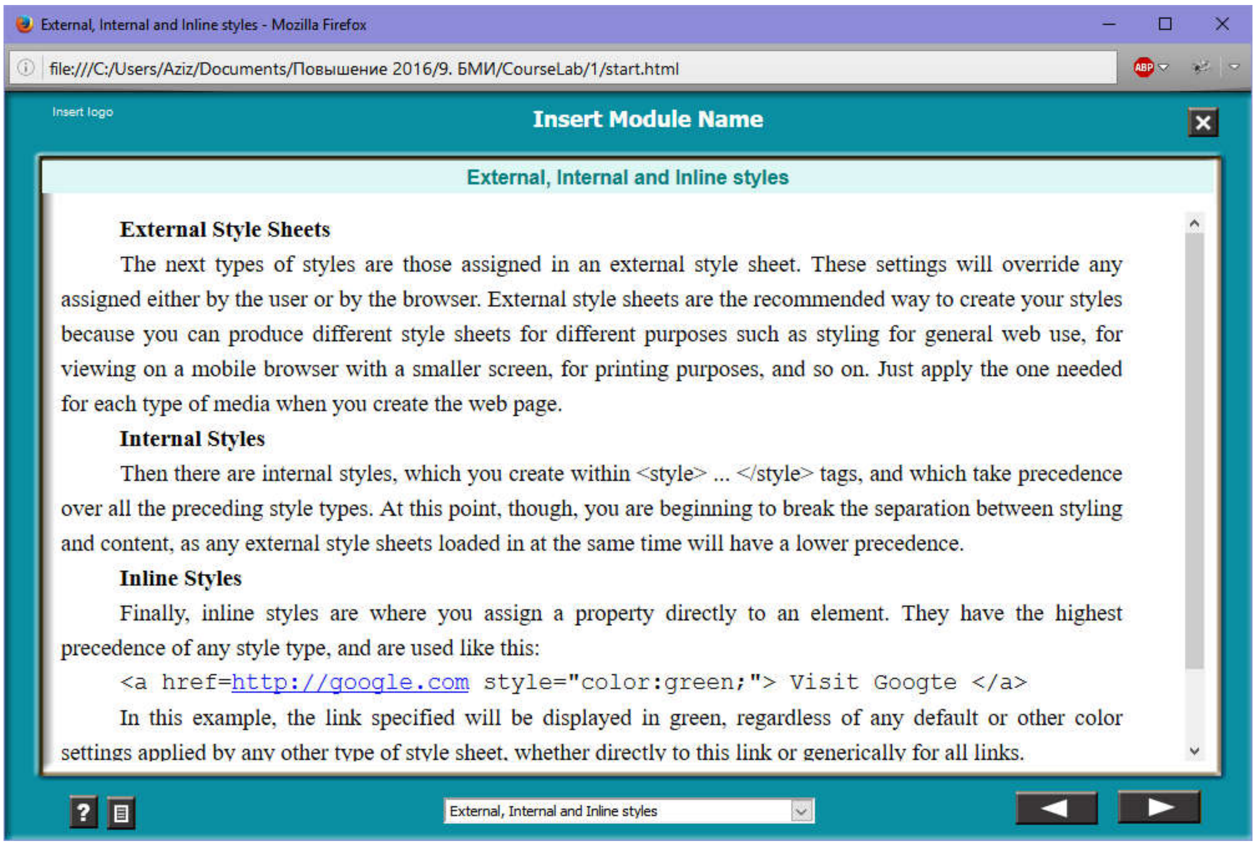
The lowest level of style precedence is the default styling applied by a web browser. These styles are created as a fallback for when a web page doesn't have any styles, and they are intended to be a generic set of styles that will display reasonably well in most instances.

Pre-CSS, these were the only styles applied to a document, and only a handful of them could be changed by a web page (such as font face, color, and size, and a few element sizing arguments).

User Styles

These are the next highest precedence of styles, and they are supported by most modern browsers but are implemented differently by each. If you would like to learn how to create your own default styles for browsing, use a search engine to enter your browser name followed by "user styles" (e.g., "Firefox user styles" or "Opera user styles") to find out how. Figure 2-2 shows a user style sheet being applied to Microsoft Internet Explorer.

If a user style is assigned that has already been defined as a browser default, it will then override the browser's default setting. Any styles not defined in a user style sheet will retain their default values as set up in


External, Internal and Inline styles - Mozilla Firefox
file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html
Insert logo
Insert Module Name

External, Internal and Inline styles

External Style Sheets

The next types of styles are those assigned in an external style sheet. These settings will override any assigned either by the user or by the browser. External style sheets are the recommended way to create your styles because you can produce different style sheets for different purposes such as styling for general web use, for viewing on a mobile browser with a smaller screen, for printing purposes, and so on. Just apply the one needed for each type of media when you create the web page.

Internal Styles

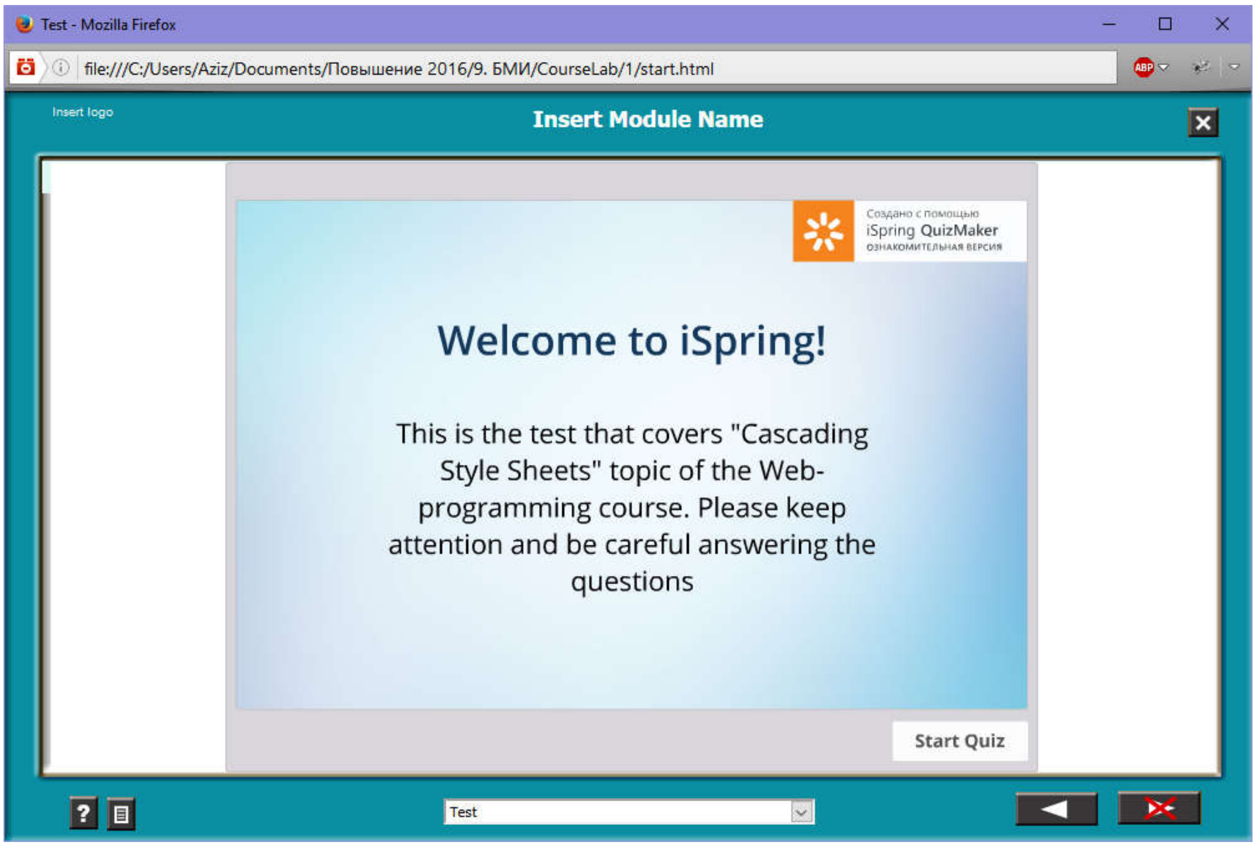
Then there are internal styles, which you create within <style> ... </style> tags, and which take precedence over all the preceding style types. At this point, though, you are beginning to break the separation between styling and content, as any external style sheets loaded in at the same time will have a lower precedence.

Inline Styles

Finally, inline styles are where you assign a property directly to an element. They have the highest precedence of any style type, and are used like this:

```
<a href=http://google.com style="color:green;"> Visit Googte </a>
```

In this example, the link specified will be displayed in green, regardless of any default or other color settings applied by any other type of style sheet, whether directly to this link or generically for all links.



Test - Mozilla Firefox

file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html

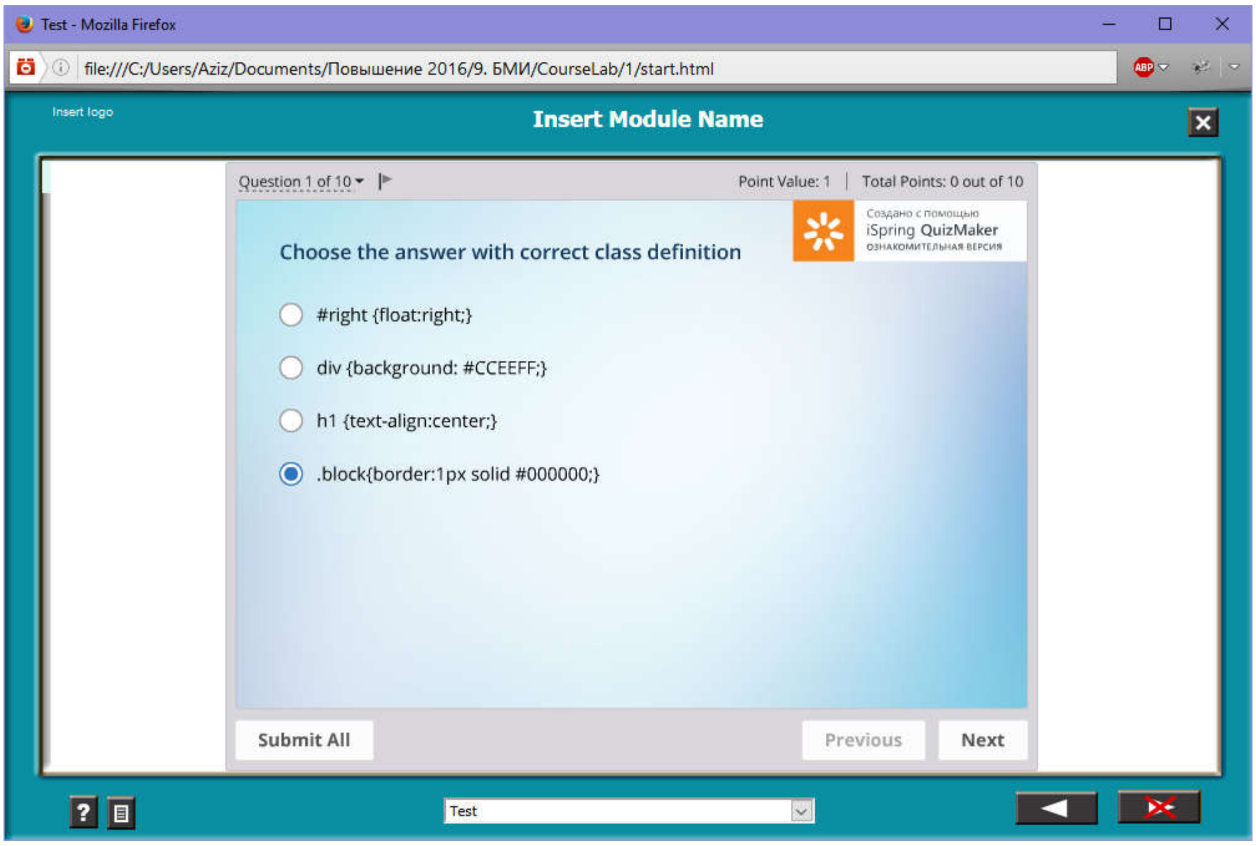
Insert logo

Insert Module Name

Создано с помощью iSpring QuizMaker ОЗНАКОМИТЕЛЬНАЯ ВЕРСИЯ

Welcome to iSpring!

This is the test that covers "Cascading Style Sheets" topic of the Web-programming course. Please keep attention and be careful answering the questions



Test - Mozilla Firefox

file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html

Insert logo

Insert Module Name

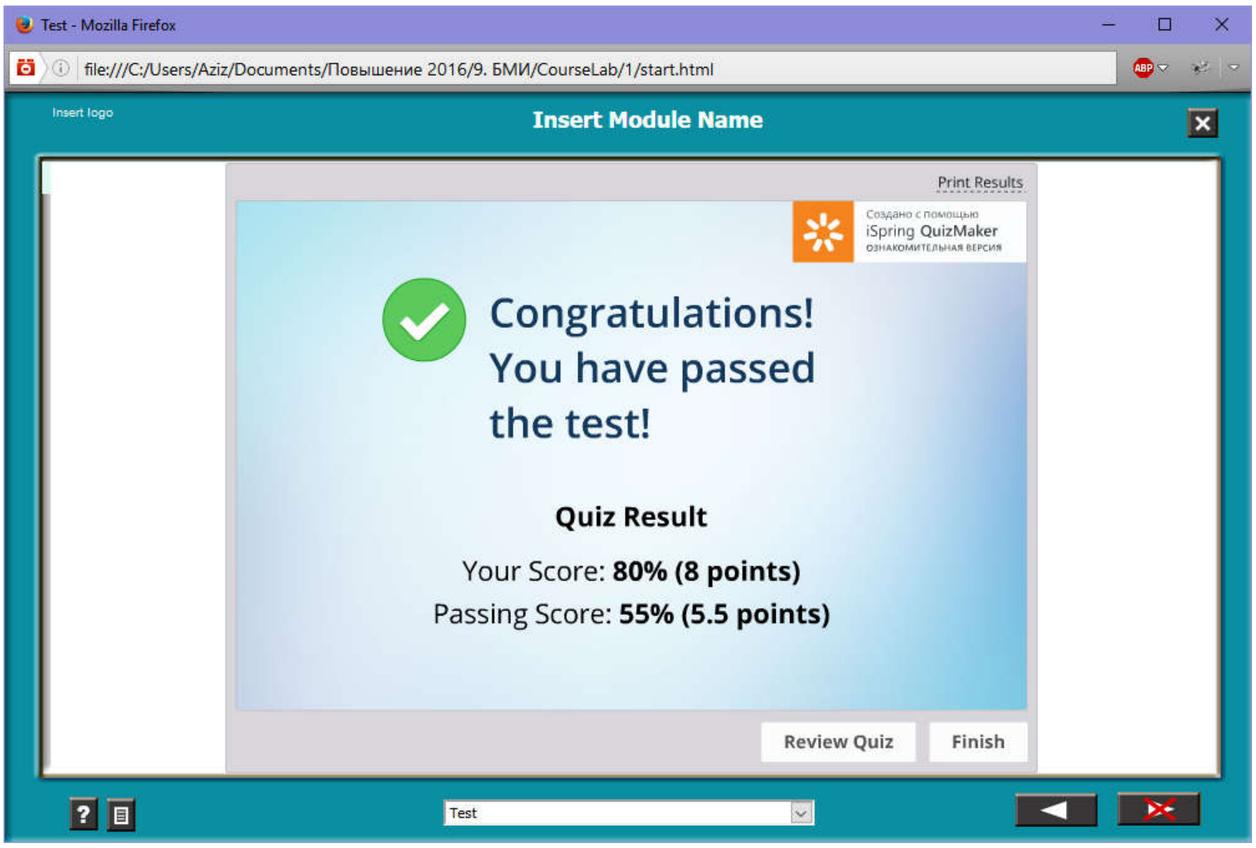
Question 1 of 10

Point Value: 1 | Total Points: 0 out of 10

Создано с помощью iSpring QuizMaker ОЗНАКОМИТЕЛЬНАЯ ВЕРСИЯ

Choose the answer with correct class definition

- #right {float:right;}
- div {background: #CCEEFF;}
- h1 {text-align:center;}
- [✓ .block{border:1px solid #000000;}]



Test - Mozilla Firefox

file:///C:/Users/Aziz/Documents/Повышение 2016/9. БМИ/CourseLab/1/start.html

Congratulations! You have passed the test!

Quiz Result

Your Score: **80% (8 points)**

Passing Score: **55% (5.5 points)**