Тема 2. HTML и CSS

В результате освоения этой темы студенты должны уметь конструировать среду, в которой выполняются программы и отображаются результаты их выполнения. Для создания такой среды мы будем использовать язык **HTML**, дополненный таблицами стилей **CSS**.

Конечным результатом изучения раздела у студента будет персональный учебный сайт, состоящий из главной (ее еще называют домашней) страницы, и ещё нескольких страниц, на которых отображаются результаты выполнения упражнений пособия. В предыдущем разделе вы уже создали первый вариант этого сайта. В этом разделе вы будете развивать созданные ранее страницы, изучая возможности HTML и CSS и выполняя упражнения. Всего необходимо будет выполнить шесть упражнений.

HTML будет использоваться снова в следующих разделах, но уже после того, как вы познакомитесь с основами программирования. Там вы узнаете, что современные страницы в соответствии с действующими стандартами создаются в рамках **объектной модели документа** (**Document Object Model - DOM**), объекты которой «понимают», как язык HTML, так и современные языки программирования. DOM, также как HTML, CSS и JavaScript, соответствует стандартам ECMA – Европейской ассоциации стандартизации и должен поддерживаться всеми браузерами. В настоящем разделе мы пока ещё будем обходиться без упоминания о DOM.

Раздел состоит из семи подразделов. После усвоения каждого подраздела необходимо выполнить упражнения, помещённые в подразделе 2.8 и ответить на контрольные вопросы,

помещённые в подразделе 2.9. Перед выполнением упражнений полезно повторить и поэкспериментировать с примерами, приведёнными по тексту подраздела.

2.1. Что такое HTML и CSS ?

HTML (Hypertext Markup Language - язык разметки гипертекста) является подмножеством более сложного языка SGML (Standart Generralized Marcup Language – стандартный обобщенный язык разметки).

Создал HTML в 1992 году в Швейцарии сотрудник Европейской лаборатории физики элементарных частиц Тим Бернерс-Ли. С тех пор выпущено уже несколько версий HTML. Сейчас используются версии 4.0 и 4.01 (HTML 4.0 и HTML 4.01).

На языке HTML, при помощи набора специальных инструкций — дескрипторов и свойств (атрибутов) этих дескрипторов, создается специальный документ: web-документ, web-страница, HTML-документ (или просто: документ, страница). Web-документ можно отображать на экране монитора программами просмотра Web-страниц (обозревателями, просмотрщиками, броузерами или браузерами – browser), например, такими, как Microsoft Internet Explorer (сокращённо - MS IE5), Mozilla Firefox, Opera, Safari, Netscape Navigator и другими.

Как правило, дескрипторы состоят из пар тэгов: открывающего и закрывающего тэга (парные тэги), хотя есть дескрипторы, состоящие из одного тэга (одиночные тэги). Закрывающий дескриптор отличается от открывающего наличием косой черты (прямого слэджа). Пара таких дескрипторов образует блок или контейнер. Контейнеры могут быть вложены друг в друга, но не должны пересекаться. Документы HTML хранятся в файлах с расширениями html или htm (например: mysite.html, index.htm).

Язык HTML интенсивно развивается. Вернее, создаются новые версии, в связи с современными потребностями. Так, например, уже существует и наряду с HTML широко используется XHTML - новая редакция HTML, соответствующая синтаксическим правилам XML (eXtensible Marcup Language – расширенный язык разметки). В отличие от HTML, XHTML позволяет создавать собственные тэги и формировать структуру документа по строгим правилам. Так, например, многие ошибки HTML-документа не замечаются браузерами, да и сами браузеры могут по-разному отображать один и тот же HTML-документ. XML-парсер (предварительный интерпретатор XML-кода) не пропускает ошибок. Любой знающий HTML, при необходимости легко освоит XHTML. Документ на этом языке легко распознать по следующим правилам записи кода, которые не всегда можно соблюдать в HTML:

- все тэги XHTML-документа записываются только в нижнем регистре;
- тэг <html> записывается с атрибутом xmlns (<html xmlns=http://www.w3.org/1999/xhtml>);
- все тэги парные, открывающему тэгу должен сопутствовать закрывающий тэг. Теги, которые не имеют закрывающего тега (например, <img> или <br>) должны иметь на конце символ «/» (например, <br />);
- значения атрибутов должны быть записаны в кавычках, например, (а в HTML браузеры воспринимают и такое: <img src=my_photo.jpg alt=фото title=фото>.

  Мы будем придерживаться выше перечисленных правил языка XHTML.

CSS (Cascading Style Sheets – каскадные листы стилей) – это появившееся позже дополнение к HTML, позволяющее форматировать и компоновать содержимое web-страницы такими способами, которые недоступны при использовании лишь дескрипторов HTML.

Многие дескрипторы выполняют сами либо с помощью атрибутов те же задачи, что и листы стилей, однако согласно действующему стандарту HTML использование таких дескрипторов и атрибутов нежелательно, так как в будущем такое написание не будет поддерживаться. С другой стороны в Интернет имеется ещё много сайтов, не переведенных на использование CSS, поскольку эта технология появилась уже после их создания.

При создании новых сайтов разработчики стремятся не нарушать рекомендуемые стандарты. Кроме этого, CSS надо знать, поскольку это – составная часть еще одного стандарта - динамического HTML (см. соответствующий раздел настоящего пособия).

CSS развивает язык HTML, позволяя отделить средства форматирования и компоновки web-страницы от содержания.

С введением технологии CSS появилась возможность избежать многократного повторения правил форматирования в каждом отдельном дескрипторе, сосредоточив эти описания в одном месте. Корректировка какого-либо правила форматирования приводит к изменению свойств сразу многих дескрипторов, связанных с этим правилом. В результате существенно повышается эффективность работы над дизайном сайта.

Кроме этого, физическое разделение файлов с информационным содержанием и стилевым оформлением дает возможность получить еще два важных преимущества:

1) повысить производительсть труда при разработке систем, поскольку теперь могут работать независимо одновременно две категории соответствующих специалистов;

2) увеличить скорость загрузки страниц, так как используемые всеми страницами одного сайта

внешние стили могут быть загружены в кэш один раз и оттуда извлекаться для каждой вновь загружаемой страницы, использующей те же самые таблицы стилей.

2.2. Основные дескрипторы и CSS-стили

2.2.1 Основные дескрипторы

Современные браузеры могут отображать на экране любой текст, однако **три дескриптора-контейнера**, как правило, присутствуют в тексте HTML-кода web-страницы для его правильной интерпретации браузером:

```
<HTML><HEAD></HEAD><BODY>содержание(тело)страницы /BODY></HTML>
```

Например, если вы загрузите в браузер такой текст:

```
<HTML><HEAD></HEAD><BODY>”Hello World”</BODY></HTML>,
```

то на экране отобразится фраза ”Hello World”, с которой, как правило, начинается программирование на любом языке. Конечно, предварительно этот текст нужно набрать в редакторе (рекомендуется Gedit, но можно в Блокноте, только не забудьте про кодировку - UTF-8) и сохранить под любым именем (рекомендуется латиница) с расширением html или htm .

Блок (контейнер), отмеченный **дескрипторами или тэгами** `<html>` и `</html>` является самым внешним и обозначает, что это документ в формате HTML. В него вложен блок `<head>...</head>`, предназначенный для размещения неотображаемой служебной информации о странице и дополнительных сведений о ней. Затем следует блок `<body> ...</body>`, содержащий всё отображаемое на экране.

Эти три основные блока можно усложнять, вкладывая в каждый принадлежащие им дополнительные блоки. Например, в блок `<head>` … `</head>` после открывающей части вставляют блок заголовка окна `<title>` … `</title>`, отображаемого браузером в самой верхней строке экрана, а в блоке `<body>` … `</body>` используются тэги для форматирования символов, абзацев и заголовков. Например: тэгами `<h1>` … `</h1>`, ..., `<h6>` … `</h6>` форматируют размер шрифта заголовков и подзаголовков, выделяя их полужирным стилем, отделяя пустыми строками и выравнивая влево; тэгами `<p>` … `</p>` - выделяют абзац текста, отделяя его пустыми строками и выравнивая влево; одиночный тэг `<br/>` - означает переход к новой строке (обратите внимание, что это один из немногих одиночных дескрипторов, обычно дескрипторы – парные).

Внутри открывающих дескрипторов можно вводить **параметры** или **атрибуты**, меняя свойства соответствующих контейнеров. Например, атрибут `bgcolor` тэга `<body>`, записанный в виде `<body bgcolor=”black”>`, меняет цвет фона на черный (по умолчанию – цвет белый, “`white`”). Здесь “`black`” и “`white`” – цвета шестнадцатиричной палитры. Добавив атрибут `text`

можно изменить цвет всего текста в теле страницы, например, <body text=”red”> изменит цвет текста в теле страницы на красный (по умолчанию цвет – черный). Умения пользоваться описанными дескрипторами достаточно, чтобы создавать простейшие сайты, состоящие из одной страницы. Мы, как упоминалось выше, **не будем пользоваться атрибутами, а будем использовать для этой цели стили**.

Тэги можно писать в одной строке или в нескольких, делая переносы в любом месте, так как служебный символ перевода строки и возврата каретки, вставляемый вашим редактором при нажатии клавиши Enter, не воспринимается браузером как разделитель тэгов.

Итак, блок <head> и </head> называется **заголовком** и является служебным блоком. Элементы этого контейнера (за исключением того, что расположено в <title> и </title>) не видны пользователю. Для обеспечения служебных возможностей в этом блоке используется одиночные тэги META. Вы уже использовали один из таких тэгов для указания кодировки UTF-8. Другие тэги META не нужны для простых страниц, однако следует помнить об их существовании, поскольку они могут потребоваться, когда вы решитесь разместить свой сайт на веб-сервере. В заголовке также можно определить связь с другими файлами (тэг LINK), например установить связь с файлом стилей потребуется в подразделе 2.2 и для выполнения упражнений.

Блок <body> и </body> идет после заголовка и описывает «тело» страницы, в котором, фактически, заключено все содержимое, отображаемое браузером. В блоке <body> и </body> в свою очередь может быть вложено очень много тэгов описания разнообразных (в том числе мультимедийных) элементов страницы. Все они подробно описаны в большом числе справочных пособий. Ссылки на рекомендуемые источники приводятся по ходу изложения и в списке литературы, однако в настоящем разделе пособия будет использоваться небольшое количество стандартных тэгов, на примере которых мы изучим основные принципы создания веб-страниц, после чего вы сможете пользоваться справочниками. Большая часть тэгов, используемых в этом разделе, приведена в табл. 2.1.

Таблица 2.1

Наиболее распространенные HTML-дескрипторы

Дескриптор	**Описание**
и … и	Шесть дескрипторов для форматирования заголовков в теле документа.
и	Разбиение текста на абзацы
 	Принудительный разрыв строки
<nobr> и </nobr>	Запрет разрыва строки

	Горизонтальная разделительная линия
<pre> и </pre>	Сохранение предварительного форматирования текста
и	Выравнивание и стилевое оформление части документа, содержащего различные блоки
 и 	Вставка ссылки
	Вставка изображения
и	Преобразует строки текста с тегами <LI> в нумерованный список
и	Преобразует строки текста с тегами <LI> в маркированный список
	Обозначает один элемент списка внутри блоков, обозначенных тегами <OL> и <UL>
<table> и </table>	Начало и окончание таблицы
<tr> и </tr>	Внутри блока TABLE создаёт строку
<td> и </td>	Внутри блока TR создаёт ячейку
<form action="url" metod="метод" name="имя" id="имя"> и </form>	Начало и окончание формы
<input type="text" value="значение" name="имя" id="имя" />	Внутри блока FORM создаёт поле для ввода строки
<input type="button" value="значение" name="имя" id="имя" onclick="url" />	Внутри блока FORM создаёт кнопку для вызова программы (скрипта) на стороне клиента
<!-- и -->	Внутри блока пишутся комментарии

Как уже говорилось, начальные тэги могут содержать атрибуты (свойства, параметры), которые определяют свойства соответствующих блоков и содержащихся в них дочерних блоков или, наоборот, переопределяют переданные по наследству свойства родительских блоков. Этих свойств также очень много и мы не будем повторять здесь справочные руководства, а рассмотрим использование некоторых атрибутов на примере тега <BODY>, задающих наиболее общие свойства документа. Многие из этих свойств можно затем переопределять в других тегах, вложенных в тело документа, дочерних по отношению к <BODY>. В табл. 2.2 приведены часто употребляемые

атрибуты блока BODY.

Внимание! Перечисленные в таблице 2.2 и все другие (не перечисленные здесь) **атрибуты** форматирования мы использовать не будем. Вместо них мы будем использовать CSS-стили (см. следующий пункт этого подраздела). Об атрибутах следует просто знать, поскольку по старинке они всё ещё используются и встречаются в Сети.

Таблица 2.2

Стандартные атрибуты

Атрибут	**Назначение**
ALINK=”цвет”	Определяет цвет активной ссылки
VLINK=”цвет”	Определяет цвет ранее посещенной ссылки
LINK=”цвет”	Определяет цвет ссылки, которая еще не посещалась
ALIGN=”center” | ”left” | ”right”	Управляет размещением текста
BACKGROUND=”URL_фона”	Указывает URL изображения, который следует использовать в качестве цвета фона документа
BGCOLOR=”цвет”	Определяет цвет фона документа
BGPROPERTIES=FIXED	Изображение фона не будет прокручиваться вместе с документом.
LEFTMARGIN=”целое число”	Определяет ширину левого поля в пикселях
RIGHTMARGIN=”целое число”	Определяет ширину правого поля в пикселях
TOPMARGIN=”целое число”	Определяет ширину верхнего поля в пикселях
BOTTOMMARGIN=”целое число”	Определяет ширину нижнего поля в пикселях
TEXT=”цвет”	Определяет цвет текста

При создании документа вначале обычно определяют цвета, которые затем будут использоваться для различных блоков текста. Если цвет не будет указан, то будут применены цвета, заданные в браузере по умолчанию либо пользователем – читателем вашей страницы, при настройке свойств обозревателя.

Цвета в HTML задаются с помощью шестнадцатиричной системы кодирования (цветовой RGB-модели), в которой используются три компоненты: красная (Red), зеленая (Green) и синяя (Blue). Каждый из этих трех компонентов соответствует шестнадцатиричному числу от 00 до FF (или от 0 до 255 в десятичной системе счисления, поэтому цвета еще можно задавать и тройкой десятичных чисел, разделенных точками). Эти три значения объединены в одно (триплет), а

впереди ставится знак #. Цвета имеют также стандартные словесные названия. В табл. 2.3 приводятся названия и коды некоторых цветов шестнадцатиричной палитры.

Таблица 2.3

Цвета шестнадцатиричной палитры.

Цвет	**Словесное значение**	**RGB-значение**
Черный	BLACK	#000000
Темно-бордовый	MAROON	#800000
Зеленый	GREEN	#008000
Оливковый	OLIVE	#808000
Темно-синий	NAVY	#000080
Фиолетовый	PURPLE	#800080
Зеленовато-голубой (электрик)	TEAL	#008080
Серый	GRAY	#808080
Серебристый	SILVER	#C0C0C0
Красный	RED	#FF0000
Лимонный	LIME	#00FF00
Желтый	YELLOW	#FFFF00
Синий	BLUE	#0000FF
Фуксия	FUCHSIA	#FF00FF
Морская волна	AQUA, CYAN	#00FFFF
Белый	WHITE	#FFFFFF

Словесные описания и коды других RGB-цветов можно найти в многочисленных справочниках, например, в размещённом на сайте RosDesign.com: http://rosdesign.com/design/politraofdesign.htm, где приведены цвета так называемой «безопасной» 216-ричной палитры. Безопасной ее называют потому, что при использовании подобранных в ней цветов все они будут более или менее хорошо различаться на недорогих мониторах пользователей, поддерживающих всего 16 цветов.

Об атрибутах для определения цветов и большинства элементов форматирования страниц более не будем упоминать, а знать вам об их существовании необходимо лишь потому, что они ещё используются на устаревших веб-страницах. Мы же будем учиться использовать не атрибуты, а

таблицы или листы стилей **CSS (Cascading Style Sheets)**.

2.2.2 Таблицы стилей

Каскадные таблицы стилей (от англ. Cascading Style Sheets, сокращённо – CSS) часто называют просто: таблицы стилей, или листы стилей, или CSS-стили.

CSS – **это набор правил форматирования тэгов HTML.** Каждое правило CSS состоит из двух частей. Первая часть называется **селектором** и в наиболее простом случае является любым HTML-тэгом, для которого во второй части - **определении**, записываются в фигурных скобках **свойства** и их **значения**, разделенные двоеточием. Синтаксис написания правил можно записать так:

селектор {свойство: значение; свойство: значение; …}
или пока что можно обойтись такой записью:
наименование_тэга {свойство: значение; свойство: значение; …}

Существует несколько методов задания таблиц стилей для HTML-документа.

В этом разделе научимся использовать листы стилей, присоединяемые к странице с помощью тэга <link …/>.

Ниже, в листинге 2.1 приведён пример html-кода из файла index.html, который вы создали при выполнении упражнения из предыдущего раздела. Лишь одна новая строка добавлена в листинге 2.1 — строка присоединяющая листы стилей из файла Petrov.css.

Листинг 2.1

```
<html>
<head>
<title>Сайт студента Петрова (версия №1)</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="Petrov.css" />
</head>
<body>
<h3>КГМТУ</h3>
<h2>Специальность "Водные биоресурсы"</h2>
<h2>Группа 10КМК</h2>
<h1>Студент Петров Иван</h1>
<h3>(учебный сайт)</h3>
<img id="my_logo" src="./img/my_photo.jpg" alt="Фото Петрова"
title="Фото Петрова" />
<p>
<a href="./exercises/exercises.html">Упражнения</a>
</p>
<h5>Керчь&copy;2010</h5>
</body>
</html>
```

В коде листинга 2.1 строка, присоединяющая листы стилей выглядит так:

```
<link rel=”stylesheet” type=”text/css” href=”Petrov.css” />
```

Это тэг LINK стандартного вида, который нужно помещать в блоке HEAD перед закрывающим тэгом </head>. В тэге LINK вам надо менять в атрибуте `HREF` путь к файлу, содержащему листы стилей. Таким образом, разработав один раз стили и поместив их в отдельный файл, вы затем можете их использовать для всех (или части) страниц сайта, помещая на каждой странице тэг LINK и указывая соответствующий путь к файлу со стилями с помощью атрибута HREF.

В файле (говорят часто - css-файле), присоединяемом в тэге LINK, в данном случае это файл `Petrov.css`, должны быть сохранены правила стилей, написанные в соответствии с приведённым выше синтаксисом. Создавать такой файл можно с помощью редактора Gedit или Блокнота. Пример некоторых возможных стилей для html-кода из листинга 2.1 приведён в листинге 2.2.

Листинг 2.2

```
body {
text-align: left; /* right/center/justify */
background-color: #F0F0DF;
color: #676767;
font-size: 24px; /* pt/em/in/mm/cm/% */
font-style: normal; /* normal/oblique/italic */
font-weight: normal; /* bold/bolder/lighter или 100/200/300/.../900 */
}
ol {
background-color: #FFFFCC;
}
```

В примере заданы правила (говорят часто - css-правила) для двух тэгов: <body>...</body> и <ol>....

Как видно из примера, наименования тэгов в листах стилей пишутся без угловых скобок. Обратите внимание: справа между парами символов /*...*/ вставлен произвольный текст с напоминанием о других возможных значениях свойств стилей. Между символами /*...*/ в листах стилей можно писать **комментарии** — текст, который игнорируется браузером. Этот текст вы можете не набирать, ошибки не будет. В данном случае комментарии используются в качестве «шпаргалки» для напоминания о других возможных значениях свойств стилей.

Поясним стили, приведённые в примере:

- `text-align: left;` - задаёт выравнивание текста, в данном случае по левому краю. Кроме этого, возможны значения: `right` — выровнять по правому краю; `center` — выровнять по центру; `justify` — выровнять по ширине;

- `background-color: #F0F0DF;` - задаёт цвет фона страницы;
- `color: #676767;` - задаёт цвет шрифта текста;
- `font-size: 24px;` - задаёт высоту шрифта текста в пикселах. Можно использовать также другие единицы измерения. Наиболее часто используемые единицы приведены в комментарии;
- `font-style: normal;` - задаёт начертание шрифта. В данном случае — шрифт обычный. Возможны также значения: `oblique` - наклонный и `italic` – курсив;
- `font-weight: normal;` - задаёт толщину шрифта. В данном случае — шрифт обычный. Возможны также значения: `bold` - полужирный; `bolder` — жирный и `lighter` — светлый (тонкий). Вместо слов можно использовать числа: 100, 200, …, 900. Каждое следующее число соответствует шрифту большей толщины. Значение 400 близко к `normal`, а 700 - к `bold`.

В листинге 2.2 заданы два селектора (стили для двух тэгов): body и ol. Для дочернего тэга ol задан свой цвет фона, который заменит цвет фона, заданный для родительского тэга body. В результате, при присоединении файла Petrov.css к странице, имеющей тэг <ol>..., цвет фона нумерованного списка будет отличаться от цвета фона остальной части страницы. Для вашего сайта, созданного при выполнении упражнения 1 — это страница со списком контрольных вопросов (файл exercises.html).

Для закрепления изложенного выше материала необходимо перейти в подраздел 2.8 «Упражнения» и выполнить упражнение 2.1 **.**

2.2.3 Стили блока

Далее Вам предстоит научиться использовать:

- стили, описывающие HTML-дескрипторы, исходя из представления HTML-дескрипторов в виде прямоугольных блоков (стилей блочной модели);
- классы стилей и имена (идентификаторы) стилей;
- стили блочной модели, а также классы и идентификаторы стилей применительно к тэгу ... .

После этого Вам необходимо выполнить ещё одно упражнение (упражнение 2.2 **в подразделе 2.8 «Упражнения»**).

Использование стилей блочной модели даёт возможность определять для HTML-дескрипторов высоту (**height**), ширину (**width**), внешние поля **(margin)**, рамки **(border)** и и внутренние поля (поля между рамкой и содержимым) **(padding)**.

Итак, блок — это прямоугольная область, описываемая некоторым дескриптором и состоящая из четырёх частей:

- содержимое элемента (контент);
- рамка (от англ. "border" — бордюр);
- внутреннее пространство - между содержимым элемента и бордюром (англ. "padding");
- внешнее пространство - между бордюром и невидимой границей прямоугольника (англ. "margin").

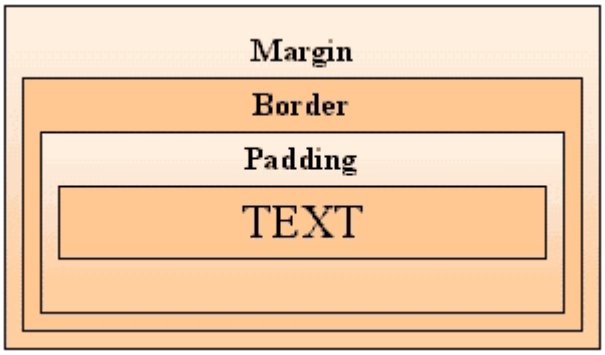

Рис. 2.1. Схема блока HTML-элемента.

На рисунке 2.1 под словом «TEXT» подразумевается некоторый фрагмент веб-документа, который может быть отдельным элементом (заголовком, абзацем, рисунком), или содержать вложенные элементы (заголовки, абзацы, рисунки и прочее).

Поясним свойства стилей блочной модели с использованием примеров:

- **margin** - установка ширины внешних полей сразу для всех сторон элемента. У этого свойства может быть от одного до четырех значений. Если имеется только одно значение, то оно будет присвоено сразу ко всем полям. Если два значения, то первое из них присваивается верхнему и нижнему полю, а второе - левому и правому. Если же три, то первое значение присваивается верхнему полю, второе - левому и правому, а третье — нижнему. Например:

  p {margin: 5% 10px 15px 2.5%}
- **padding** — установка ширины внутренних полей сразу для всех сторон элемента аналогично тому, как это делается для свойства margin. Например:
  p {padding: 20px 10px 15px 25px}
- **border** — установка рамки сразу со всех четырёх сторон элемента. Установка рамки делается немного не так, как для свойств margin и padding. Для рамки необходимо задать ширину, стиль (сплошная, прерывистая, двойная и т. п.) и цвет. Например:
  img {border: 1px, solid, blue} — задаёт вокруг изображения тонкую (1px), сплошную (solid), голубую (blue) рамку.

Можно также устанавливать свойства для каждой из сторон блока. Все свойства стилей блочной модели можно взять из многочисленных справочников. Вы должны научиться

пользоваться стилями блочной модели, например, по рекомендуемым справочным пособиям перечисленные в конце этого раздела (2.10.Справочные пособия).

Однако прежде, чем использовать стили блочной модели, познакомимся ещё с одним HTML-дескриптором, а именно — дескриптором ... .

**Дескриптор ... ** позволяет объединять в виде блока фрагменты HTML-кода, содержащие другие дескрипторы. Поэтому использование стилей, описывающих блок, является наиболее эффективным и показательным именно для дескриптора ... , хотя их, конечно, можно использовать для различных тэгов.

Дескриптор ... совместно со стилями блочной модели широко используется для вёрстки веб-страниц. Приведём преобразованный пример HTML-кода из предыдущего раздела:

Листинг 2.3

```
<html>
<head>
<title>Сайт студента Петрова (версия №1)</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="Petrov.css" />
</head>
<body>
<div id="header">
<a class="headerlink" href="../../../../../">КГМТУ</a>
<a class="headerlink" href="../../../groups/?cod=10kmk" >Группа</a>
</div>
<div id="container">
<div id="content">
<h1 id="logo">Петров Иван Петрович</h1>
<h2>О себе</h2>
<p>Родился и вырос в г.Керчь (АР Крым). Живу в р-не завода
им.Войкова, недалеко от раскопок древнегреческого
поселения Мирмекий.
Окончил школу им.Веры Белик в 2010 году. Сейчас - студент КГМТУ.
Учусь по специальности - "Водные биоресурсы и марикультура".
</p>
<h2>Интересы</h2>
<p>Люблю узнавать новое, ранее неизвестное, из истории Керчи.
И, конечно, связанное с морем и моей будущей специальностью.
Вот, к примеру!
Куда Человечество в лице Гомера отправило экспедицию Одиссея?
</p>
<p>Или ... - можно ли считать целью одной из экспедиций
аргонавтов одно лишь руно, хоть и золотое?
О чём, более важном, в своей поэме хотел рассказать
потомкам Аполлоний Родосский?
</p>
<p>
<a class="extlink" href="./hobby/argo.html">Продолжение ...</a>
</p>
<h2>Программирование</h2>
<p>Для того, чтобы я смог в будущем проводить эксперименты с
```

```
изучаемыми объктами марикультуры,
мне необходимо овладеть программированием. Ниже я разместил
ссылку на некоторые из моих
упражнений по программированию.</p>
<p class="bottom">
<a class="extlink" href="./exercises/exercises.html">Упражнения …
</a>
</p>
</div>
</div>
<div id="footer">Петров&copy;2010</div>
</body>
</html>
```

Стили, соответствующие HTML-коду листинга 2.3, приведены в листинге 2.4.

Листинг 2.4

```
body {
text-align: center;
font-family: Verdana, Tahoma, Arial, serif;
font-size:100%; /* base font size is 16px */
background: #999999 url('./img/back.jpg') repeat-x fixed top center;
margin: 0em;
color: #676767;
}
#container {
width: 59.5em; /* около 952px */
background-color: #f0f0df;
text-align: left;
margin-left: auto;
margin-right: auto;
margin-top: 0;
margin-bottom: -0.8em;
}
#header { /* Header Formatting */
background: #999999 url('./img/back.jpg') repeat-x top center;
}
.headerlink {
background: white;
padding-right: 24px;
color: silver;
margin: 0 0.1em;
}
#content {
font-size: 1em;
font-family: Tahoma, sans-serif;
margin: 0em 1em;
}
img#my_logo {
float: left;
border: 0;
}
#content h1,h2 {
font-family: Tahoma, sans-serif;
color: #776157;
font-weight: bold;
```

```
}
#content h1#logo {
background-image: url(./img/my_photo.jpg);
background-repeat: no-repeat;
background-position: top left;
text-align: center;
vertical-align: top;
font-size: 3em;
margin-top: 0.1em;
margin-bottom: 0.5em;
padding-top: 0.7em;
padding-bottom: 0.6em;
}
#content h2 {
font-size: 1.5em;
margin-top: 0.25em;
margin-bottom: 0.25em;
border-bottom: 1px solid #154ead;
}
p.bottom {
padding-bottom: 1em;
}
a {
color: #776157;
text-decoration: none;
}

.extlink {
background: white;
padding-right: 24px;
font-style: italic;
}
ol {
background-color: #ffffff;
font-size: 14px;
border-style: dotted;
border-width: 1px;
}
#footer {
text-align: center;
font-size: 0.75em;
color: #ffffcc;
padding: 0 0 1em 0;
}
```

Код из листинга 2.3, после присоединения стилей из листига 2.4, в браузере будет иметь вид, показанный на рис. 2.2.

Рис.2.2. Вид веб-документа, соответствующего HTML-коду и стилям из листингов 2.3 и 2.4.

Поясним CSS-стили листинга 2.4 и как они применяются в HTML-коде листинга 2.3 (только те CSS-стили, которые Вам придётся применять при выполнении упражнений).

1) CSS-стиль (см. листинг 2.4)

```
#container {
width: 59.5em; /* около 952px */
background-color: #f0f0df;
text-align: left;
margin-left: auto;
margin-right: auto;
margin-top: 0;
margin-bottom: -0.8em;
}
```

создаёт стиль с идентификатором (именем) `#container`. Этот стиль можно использовать для форматирования различных тэгов. Для использования стиля в открывающем тэге соответствующего дескриптора с помощью атрибута ID указывается идентификатор (см. листинг 2.3):

```
<div id="container">
```

Аналогичным образом созданы CSS-стили с идентификаторами: `#header`, `#content` и `#footer` используются для стилевого оформления заголовка сайта, его содержания и концовки (подвала)

сайта (найдите в листинге 2.3 тэги, в которых указаны эти идентификаторы и разберитесь, какие эффекты форматирования они вызовут).

2) CSS-стиль:

```
img#my_logo {
float: left;
border: 0;
}
```

создаёт стиль с идентификатором #my_logo, который можно использовать только лишь для тэгов <img ...>. Для использования стиля в тэге <img ...> с помощью атрибута **ID** указывается идентификатор (в листинге 2.3 тэга <img ...> нет, попробуйте применить его самостоятельно):

```
<img id="my_logo" src="...">
```

3) CSS-стиль (см. листинг 2.4):

```
.headerlink {
background: white;
padding-right: 24px;
color: silver;
margin: 0 0.1em;
}
```

создаёт класс, который можно использовать для форматирования различных тэгов. Для использования стиля в открывающем тэге соответствующего дескриптора с помощью атрибута **class** указывается имя класса (см. листинг 2.3):

```
<a class="headerlink" href="../../../../../">КГМТУ</a>
```

4) CSS-стиль (см. листинг 2.4):

```
p.bottom {
padding-bottom: 1em;
}
```

создаёт класс только лишь для тэга ... (см. листинг 2.3):

```
<p class="bottom">
```

2.2.4. Специальные символы

И в завершение этого подраздела — о **специальных символах**.

Иногда вам понадобится вставлять в текст символы, подобно тому, как вы это делаете с помощью меню редактора MS WORD (Вставка-Символ). В HTML это делается очень просто. Например, если вам нужен символ π (математический символ числа пи), используйте его

заменитель π (точку с запятой надо ставить обязательно). Список этих заменителей символов довольно велик и их можно найти в большом числе руководств, в том числе на странице `http://www.w3.org/TR/WD-entities`, а в табл. 2.4 приведены заменители символов, используемые в настоящем пособии.

Таблица 2.4

Наиболее распространенные специальные символы

Символы	**Именованные заменители**	**Цифровые заменители**
"	&quot;	&#34;
&	&amp;	&#38;
<, >	< >	< >
$		&#036;
@		&#064;
π	&pi;	
©, ®	© ®	© ®
±	&plusmn;	&#177;
²	&sup2;	&#178;
³	&sup3;	&#179;
º	&deg;	&#176;
·	&middot;	&#183;
¼, ½, ¾, ⅓, ⅔, …	¼ ½ ¾	¼ ½
÷	&divide;	&#247;
≈	&asymp;	&#8776;
≠	&ne;	&#8800;
≤	&le;	&#8804;
≥	&ge;	&#8805;
∞	&infin;	&#8734;

Например, строка `Петров&copy;2010` будет отображена в браузере как: Петров©2010. Если вы внимательно просмотрели листинг 2.1, то увидели, что там уже стоит такая строка вместо той, что была раньше (Керчь-2010).

Изложенных здесь понятий достаточно, чтобы создавать собственные странички, пусть и простые, но уже такие, которые вы можете использовать для размещения статической информации в Интернет (вспомним: статическими называют страницы, не меняющиеся без ручной переделки

HTML-кода человеком, в отличие от динамических, когда весь HTML-код или его часть изменяется программным способом, соответственно обновляя вид и содержимое страницы, отображаемой в браузере).

Вам должны быть понятны принципы написания простых веб-страниц и их стилевого оформления, а подробные описания приведённых в примерах тэгов и стилей можно найти в многочисленных справочниках и учебниках, которые, в свою очередь можно найти с помощью поисковых сервисов, вводя соответствующие строки запросов. Ниже приведены ссылки на рекомендуемые справочники и учебники:

1. HTML-справочник (наиболее простой) — http://analog.com.ua;
2. CSS-справочник (наиболее простой) — http://css.manual.ru/;
3. HTML/CSS/...- справочник более сложный — http://stepbystep.htmlbook.ru/;
4. Лекции по таблицам стилей (CSS) — http://www.intuit.ru/department/internet/css/1/;
5. Курс обучения использованию CSS — http://www.intuit.ru/department/internet/css/;
6. RGB-цвета безопасной палитры: http://rosdesign.com/design/politraofdesign.htm .

Для закрепления материала необходимо перейти в подраздел 2.8 и выполнить два упражнения: упражнение 2.1 **и** упражнение 2.2**.**

2.3. Ссылки

Следует знать о том, что главное в HTML – создание гиперссылок. Гиперссылки создаются с помощью тэга **<a>** и **** (от англ. anchor – якорь). Сослаться (перейти) к другому файлу (другой странице) можно с помощью тэга:

** *элемент привязки* **

Итак, ссылка состоит из двух различных частей:

- во-первых, это элемент привязки, то есть место в документе, над которым необходимо что-то сделать, чтобы вызвать переход. Элемент привязки в ссылке может быть словом, группой слов (текстовый тип) или изображением (графический тип).
- во-вторых, это URL, который указывает браузеру, куда ему обращаться для перехода, при щелчке на элементе привязки. При щелчке браузер загружает файл или страницу, расположенную по указанному URL.

Точно также, как и для ссылок на другие HTML-файлы, устанавливаются и элементы привязки для ссылок на разделы внутри текущего документа. Например, если документ большой, то удобно в верхней части web-страницы поместить содержание и связать его строки с соответствующими разделами web-страницы. Опишем создание таких ссылок более подробно, поскольку тут есть одна особенность.

Возможны два метода создания ссылок внутри одного и того же документа:

- по первому методу это делается с помощью не одного, как обычно, а с использованием двух контейнеров <A> …</A>. Сначала с помощью одного контейнера в HTML-файле создается элемент привязки, указывающий, куда переходить по ссылке (это аналог «закладки» в MS WORD). Такому элементу привязки присваивается имя с помощью атрибута NAME открывающего дескриптора <A>. Атрибут HREF в этом теге первого контейнера не используется, поэтому браузер не будет выделять этот контейнер как гиперссылку. Имя элемента привязки используется затем в ссылке на него в атрибуте HREF первого тэга второго контейнера. При этом перед именем ставят знак #.
- по второму методу для создания элемента привязки внутри документа вместо контейнера <A>…</A> используется атрибут ID. Атрибут ID можно использовать для присвоения имени любому HTML-контейнеру и потом ссылаться на него (осуществлять переход) по этому имени из контейнера <A> … </A> с помощью атрибута HREF, также, как и по первому методу. Далее, для создания ссылок, мы будем использовать второй метод.

Ссылки также могут быть использованы для перехода к определенным частям других документов. Предположим, вы хотите сделать ссылку из документа A на специфическую секцию документа B (назовем этот файл documentB.htm.) Во-первых, вам надо присвоить имя нужному тэгу с в документе B. Например, чтобы поименовать тэг, называющийся "example" в документе B, наберите:

```
это текст-закладка <A ID = "example">текст закладки</a>
```

Таким образом вы отметили место в документе B, на которое хотите сделать ссылку. Теперь, когда вы будете создавать ссылку из документа A, укажите не только имя файла с документом B, но и конкретное место в нем, отделив название места символом (#), например так:

```
Это - <a HREF="documentB.htm#example">ссылка</a> на текст закладки  в документе B.
```

Переход по «ссылка» в документе A отошлет читателя непосредственно к словам «текст закладки» в документе B.

Повторите этот пример на страницах своего сайта, чтобы закрепить навыки создания ссылок для перехода к нужным местам различных страниц.

Далее научимся использовать стили для создания эффектов, обычно применяемых в документах для стилевого оформления ссылок.

Это так называемые **псевдоклассы** — правила стилей, которые изменяются со временем или с помощью действий пользователя.

Синтаксис применения псевдоклассов следующий:

```
Элемент:псевдокласс { правило1_стиля: свойство стиля; ... }
```

Далее будем осваивать создание ссылок на примере с кодом страницы, на которой уже размещены различные типы действующих ссылок, а затем вам будет предложено доработать эту страницу в соответствии с упражнением 2.3.

Внимание! Пример использования листов стилей для создания меню ссылок просмотрите и разберите на сайте, созданном студентом Дудка Никита по адресу:
http://pvn.ho.ua/pvn.org.ua/www/students/10kmk183/www/index.html

Для закрепления материала необходимо перейти в подраздел 2.8 и выполнить упражнение 2.3**.**

2.4. Вставка изображений

В приведенном выше примере вам уже пришлось вставлять в документ изображения и даже использовать их для создания ссылок. Поэтому вы уже знаете, что для вставки изображений существует тэг **<img>**. Здесь мы более подробно рассмотрим возможности использования этого тэга, поскольку изображения используются в любой области деятельности. Манипулировать изображениями мы научимся на примере простого фотоальбома (далее так и будем называть этот пример). Для освоения настоящего подраздела служит типовая страницы №3 «Интересы», на основе которой необходимо создать собственную подобную страницу, на которой необходимо отобразить упомянутые выше примеры: фотоальбом, ГИС, графики. Примеры этой страницы будут затем использоваться и последовательно развиваться в процессе освоения программирования в следующих подразделах.

Что понимается под словом «изображение»? Изображение – это прежде всего файл, в котором хранится информация в графическом виде определенного формата. Обычно для web-документов используются растровые графические форматы: GIF, JPEG, PNG. Файлы с изображениями в таких форматах хранятся соответственно в файлах с расширениями: gif, jpg, png. Каждый из перечисленных форматов имеет свои достоинства и недостатки. Формат GIF поддерживает всего 256 цветов, а JPEG – 16.7 миллионов цветов, а PNG – еще больше. Поэтому для встраивания фотографий лучше использовать формат JPEG и PNG, а рисунков с небольшим количеством цветов и оттенков (логотипов, пиктограмм, схем, диаграмм) - формат GIF.

Небольшие рисунки в формате GIF наиболее часто используют для украшения страницы, поскольку он поддерживает так называемую прозрачность изображения и простую анимацию. К тому же формат GIF хорошо сжимает изображения, содержащие небольшое количество цветов, в результате чего страницы с такими изображениями быстро загружаются. В чересстрочной

технологии загрузки также используются GIF-изображения (это позволяет быстро просматривать менее качественное изображение, до загрузки всех его строк).

Вообще-то существует понятия «сжатие с потерями» и «сжатие без потерь». Так, при сохранении в формате GIF изображение сжимается без потери деталей (при небольшом количестве цветов, конечно). При сохранении в формате PNG также происходит сжатие без потерь, а JPEG – сжатие с потерями. Именно поэтому JPEG эффективно сжимает файлы. Поэтому, если размер и соответственно скорость загрузки для вас важнее, используйте JPEG.

Для предварительной подготовки изображений вам необходимо соответствующее приложение. Существует много мощных многофункциональных коммерческих графических пакетов, например, Adobe Photoshop. Однако для достижения задач настоящего конспекта достаточно возможностей более простых приложений. Вам не потребуются сложные графические пакеты для создания изображений, а необходимые операции с готовыми изображениями (уменьшение размеров, сохранение в различных форматах, вырезание прямоугольных участков и некоторые другие) можно выполнять с помощью известного стандартного графического редактора Paint. Более сложные операции с изображениями рекомендуется выполнять с помощью бесплатно распространяемого графического редактора Gimp, который можно загрузить и для Linux и для Windows, зайдя на сайт по адресу: http://www.izone.ru/graphics/editors/gimp.htm. Руководство по работе с редактором можно найти здесь: http://linux.armd.ru/common/img/uploaded/files/Gimp.pdf .

Итак, дескриптор <img ...> (от англ. image) позволяет вставить графическое изображение. Этот дескриптор – одиночный, то есть не имеет закрывающего тэга. При этом в атрибуте дескриптора **src** указывается имя и место размещения (адрес) соответствующего графического файла. Например, тэг <img src="../image1/pic1.jpg"> позволяет вставить изображение (фотографию или любое другое изображение, например, такое, как в предыдущем примере), находящееся в папке image1, которая расположена на одном уровне выше папки example_HTML, содержащей файл task_2_2_PIP.htm с кодом вашего документа. Указав в качестве адреса URL, вы предпишите загрузку соответствующего файла. Кроме атрибута src можно использовать атрибут lowsrc, задав с его помощью загрузку копии основного изображения, такого же размера, но с низким разрешением (а значит занимающего меньше памяти и быстро загружающегося), появляющегося до загрузки основного изображения, например: <img src="../image1/pic1.jpg" lowsrc="../image1/pic1.jpg">. Этот эффект используется при медленной загрузке основного изображения. В результате, пока загружается основное изображение, пользователи могут работать с вполне работоспособной страницей, временно

содержащей быстро загружающееся упрощенное изображение. Ускорения загрузки изображения можно добиться также, если указать размеры изображения, например: `<img src="../image1/pic1.jpg" style="width:100px; height:60px">`. Если размеры не указаны, то браузер отобразит страницу лишь после окончания загрузки изображений, поскольку не может заранее скомпоновать страницу без информации о размерах входящих в неё изображений. При указании размеров, браузер, зная, где будет место под изображения, подготовит макет страницы, отобразит текст и постепенно будет загружать изображения, а пользователь тем временем уже может работать с текстом и ссылками, не теряя времени из-за медленной загрузки изображений.

Если загружаемое изображение больше, чем пространство, которое вы зарезервировали с помощью `width` и `height`, то браузер сожмёт изображение, а если меньше, то увеличит изображение, подгоняя его под зарезервированное под него пространство. Естественно, что при этом могут исказиться пропорции, а при увеличении изображения еще и качество из-за проблемы растеризации растрового изображения, когда изображение становится зернистым, а его контуры – ступенчатыми из-за того, что браузер вынужден увеличивать каждый пиксел исходного изображения.

Для того чтобы можно было манипулировать совместным расположением изображения и рядом расположенного текста, для тэга `<src>` предусмотрен стиль `float`, который может принимать значения `left` (изображение будет находиться слева от текста) или `right` (изображение - справа от текста). Можно создать также пустое пространство вокруг изображения с помощью стилей margin и padding, которые вы должны были изучить при освоении таблиц стилей. Вокруг изображения можно создать рамку, если указать стиль border, например, так: border: 1px solid gray.

На тот случай, если пользователи отключили в броузере загрузку изображений в дескрипторе `<img>` предусмотрен атрибут `alt(alt="мой портрет")`. Значение этого атрибута (это обычно текст, описывающий изображение) отображается в месте, предусмотренном для изображения. Если же загрузка изображений не отключена, то поясняющий текст отображается при наведении курсора на изображение, если указать атрибут `title(title="мой портрет")`.

Изображения можно также использовать для украшения фона страницы. Такие изображения называют фоновыми изображениями и задаются они с помощью стиля `background`, например, так: `background: green URL(../image1/pic1.jpg).` При загрузке страницы вначале появляется фоновый цвет, и пользователь может работать со страницей не ожидая, пока загрузится фоновое изображение. Для того, чтобы страница сразу же имела работоспособный вид вы должны

подобрать фоновый цвет, похожий на основной цвет изображения и, чтобы виден был текст (например, на белом фоне белый текст виден не будет). Обычно подбирается фоновое изображение, имеющее небольшой размер, а броузером оно многократно повторяется, заполняя всю страницу. Можно также использовать стиль `background-propeties`, которому можно задавать значение `fixed` (`background-propeties:fixed`) и тогда фоновое изображение не прокручивается при прокручивании страницы в окне.

Еще один полезный эффект при работе с изображениями можно получить с помощью ссылок. Так, например, в качестве элемента ссылки можно указать уменьшенный вариант быстро загружающегося изображения, а в атрибуте `href` задать место размещения большого качественного, но медленно загружающегося изображения. С использованием таким образом построенных ссылок легко сделать простейший фотоальбом, когда при нажатии на миниатюрном варианте изображения в отдельном окне появляется качественное изображение. Ну и, конечно, графические изображения часто используют в ссылках.

Пример того, что вы должны уметь делать с изображениями можно посмотреть на сайте http://pvn.ho.ua, на примерах того, что сделал студент Дудка Никита (и другие, например, Жаворонкова Оля, Березовская Даша).

Это пример достаточно большого документа с несколькими абзацами, в каждом из которых вставлено по изображению, управлять которыми необходимо научиться: размещать влево-вправо от текста, чтобы изображения не сливались с текстом следующего или предыдущего параграфа, чтобы обеспечивались необходимые отступы между текстом и изображением, рамка выглядела так, как это Вам хотелось бы.

Кроме того, на этом примере нужно научиться управлять фоновыми изображениями: для страницы в целом и для блоков внутри страницы.

Некоторые полезные тонкости по формированию простейшей галереи изображений с подписями с использованием строчно-блочных элементов можно прочитать здесь: http://htmlbook.ru/samlayout/blochnaya-verstka/strochno-blochnye-elementy.

Для закрепления материала необходимо перейти в подраздел 2.8 и выполнить упражнение 2.4.

2.5. Таблицы

Освоим далее требуемые для выполнения заданий конспекта простые приемы форматирования – размещение элементов содержания страницы в нужном порядке с помощью таблиц (только основные дескрипторы) и вставки изображений.

Таблицы используются очень часто для размещения различных участков документа в

нужном месте. Такое использование таблиц часто называют табличным дизайном. Альтернативой использованию таблиц является описанная выше фреймовая технология и и пока еще не известные вам слои динамического HTML с использованием контейнера <DIV>) и </DIV> (см. следующий подраздел).

Таблицы создаются очень просто, аналогично тому, как это делается в редакторе MS Word, но только теперь вы можете управлять их созданием с помощью, вставляемых внутрь тела страницы дескрипторов: <table> .. </table>, <tr> .. </tr>, <td> .. </td>. Первый блок определяет таблицу, второй вставляется в первый и определяет строку, и, наконец, третий вставляется во второй и определяет столбец. Например, таблица с двумя строками и тремя столбцами может быть создана следующим образом:

Листинг 2.5

```
<HTML> <HEAD><TITLE> … </TITLE></HEAD>
<BODY>
<!-- далее задаётся таблица -->
<TABLE style="width:95%; height=80%; border:1px solid gray; color:green">
<TR style="color: red">
<TD>Содержание столбца 1 строки 1 </TD>
<TD>Содержание столбца 2 строки 1 </TD>
<TD>Содержание столбца 3 строки 1 </TD>
</TR>
<TR>
<TD>Содержание столбца 1 строки 2 </TD>
<TD style="color: red">Содержание столбца 2 строки 2 </TD>
<TD>Содержание столбца 3 строки 2 </TD>
</TR>
<TR>
<TD>Содержание столбца 1 строки 3 </TD>
<TD>Содержание столбца 2 строки 3 </TD>
<TD>Содержание столбца 3 строки 3 </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

В примере цвет текста, толщина рамки в пикселях, ширина и высота задаётся для всей таблицы в процентах от размеров экрана (по умолчанию рамка имеет нулевую толщину, а размеры таблицы управляются содержимым ячеек). При необходимости стили для отдельных строк и ячеек могут быть переопределены, как это делается в примере для первой строки и второй ячейки второй строки.

Для того, чтобы закрепить навыки построения html-таблиц рекомендуется повторить и поэкспериментировать с приведённым выше примером.

Более сложные примеры html-таблиц с использованием таблиц стилей можно посмотреть на

сайте pvn.ho.ua на персональных учебных сайтах, которые создали бывшие первокурсники Дудка, Петров, Березовская, Жаворонкова и др. (зайдя на их сайты, выберите страницу "Упражнения").

Внимание! Для закрепления материала необходимо перейти в подраздел 2.8 и выполнить упражнение 2.5.

2.6. Позиционирование и слои

Контейнер `<div> … </div>` можно использовать для форматирования целого блока элементов страницы. Задавая стили для этого контейнера можно выровнять, например, сразу заголовок, текст в абзаце и рисунок, как в следующем примере:

```
<div style="text-align:center; width:600px">
<h2>Текст заголовка</h2>
<p style="text-align:left"><img src="pic1.gif" style="float:right">Строки
текста первого абзаца.</p>
<p>Строки текста второго абзаца.</p>
</div>
```

В этом примере все элементы форматируются так, как указано в открывающем теге , то есть по центру, за исключением элементов, для которых указано другое значение стиля text-align (в данном случае для тэга <p>, строка текста в котором будет смещена влево и рисунка - `float:right`).

В тэге `<div>`, как и для других тэгов, можно указывать различные стили. Например, для изменения цвета фона для всех элементов выше приведенного блока в тэге `<div>` можно указать соответствующее определение стиля следующим образом:

```
<div style=”background-color:yellow”>
<h2>Текст заголовка</h2>
<img src=”pic1.gif”><br>
Это неформатированная строка текста.
<p>Это форматированная, как абзац, строка текста.</p>
</div>
```

Далее - о тэге `<div>` в связи со слоями.

Немного теории для тех, кто никогда не сталкивался с термином "слой". Слой (англ. layer) - это прямоугольный блок, который может содержать различные объекты - текст, графику, таблицы. Допускается применение нескольких слоев в пределах одного документа, которые могут взаимно перекрываться, с помощью чего можно добиваться различных интересных эффектов. Кроме того, содержимое слоя в любой момент можно отредактировать или вообще удалить, не трогая элементы других слоев. Это гораздо удобнее, чем заново создавать картинку (что приходилось бы делать при невозможности использования слоев). Слои давно уже стали привычной технологией для пользователей профессиональных графических и издательских пакетов. Слои можно создавать вручную с использованием графических редакторов типа платного Adobe Photoshop или свободно

распространяемого Gimp. Но часто лучше это делать в HTML, а затем автоматизировать изменения с помощью Javascript, например, при отображении динамически изменяемых графиков или карт.

Шаблон HTML-кода слоя выглядит так:

```
<div style="Свойства слоя">
Содержимое слоя
</div>
```

На месте строки "Содержимое слоя" находятся HTML-теги, которые определяют элементы, из которых состоит слой. Свойства слоя - это свойства таблицы стилей CSS, состоящие из нескольких пар «свойство:значение», которые отделяются друг от друга точками с запятой. Рассмотрим эти свойства:

- position:absolute

  Без указания этого свойства (именно со значением absolute) вы не сможете установить позицию слоя относительно левого верхнего угла окна броузера с помощью свойств left и top (см. ниже).

- left:число

  Определяет расстояние от левого края окна браузера до левого края слоя. Допускается указывать эту величину в разных единицах измерения. Можно подстраховаться и принудительно задать в качестве единиц измерения пиксели:

- left:50px.
- top:число

  Определяет расстояние от верхнего края окна броузера до верхнего края слоя. Все остальное - аналогично свойству left.

- width:число

  Это свойство указывает ширину слоя. Если указанная ширина слоя меньше, чем ширина картинок, таблиц и прочих неразрывных элементов, ширина слоя будет определяться наиболее широким элементом. Можно указывать ширину и в процентах (width: 80%), правда, это дает несколько странный результат в различных браузерах.

- height:число

  Определяет высоту слоя.

- z-index:число

  Это свойство служит для определения порядка расположения слоев на странице. Чем меньше значение свойства z-index, тем позднее выводится слой на экран. Например, слой со значением z-index, равным 0, будет перекрывать слой с z-index:1.

Таким образом, пример с тэгами ... , определяющими два слоя, может выглядеть так:

```
<div style="position:absolute; top:25; left:25; width:200; height:400;
background-color:#ff0000; background-image:url(./img/img1.gif); z-index:1">
Содержимое слоя 1
</div>
<div style="position:absolute; top:25; left:25; width:100; height:200;
background-color:#0000ff; background-image:url(./img/img2.gif); z-index:0">
Содержимое слоя 2
</div>
```

Кстати, если вам нужен слой, содержащий всего один объект, вы можете не создавать отдельный тег DIV, а указать атрибут STYLE прямо внутри тега, который описывает соответствующий элемент, например:

```
<img src=img1.jpg  style="Свойства слоя">
```

Спецификации CSS предусматривают значительно большее количество различных свойств, которые можно указывать в значении атрибута STYLE. Поэкспериментируйте, и, возможно, вы откроете какие-нибудь новые интересные эффекты использования слоев.

Резюме. В HTML4 слой это базовый элемент вёрстки веб-страниц, при которой активно применяются стили и придерживаются спецификаций HTML и CSS.

Следует заметить, что в HTML5 [14] добавлено несколько новых тегов разметки для обозначения разных типовых блоков страницы. Теги `<header>` и `<footer>` используются для создания «шапки» и «подвала», `<nav>` - для навигации, `<aside>` - для боковой панели со второстепенным контентом. Эти элементы предназначены для замены тега `<div>`, придавая смысл разметке. В документах на HTML5 активно применяется термин «элемент», под которым подразумевается соответствующий тег и элемент который он создаёт.

Изложенные выше принципы блочной разметки при этом сохраняются.

Для закрепления материала необходимо перейти в подраздел 2.8 и выполнить упражнение 2.6.

2.7. Формы

В связи с важностью для практических приложений рассмотрим более подробно использование форм. Формы используются для ввода пользователем данных, в том числе в интерактивном режиме, для последующей обработки этих данных программой-приложением. Для интерактивного взаимодействия с документом посредством формы используются ранее рассмотренные понятия события и обработчика события, а обработка данных осуществляется с помощью понятия функции, в том числе встроенной в качестве метода в тот или иной объект. В HTML форма создаётся с помощью тэга `<FORM>` … `</FORM>` с некоторыми атрибутами.

Элементы формы отображаются на вэб-странице посредством специальных HTML-тэгов, называемых **элементами управления**. Каждый элемент управления предназначен для передачи полученных от пользователя данных соответствующему приложению (возможно и обратное). Современные языки программирования, также как и **JavaScript**, имеют встроенные механизмы (интерфейс) передачи данных между элементами управления и программой-приложением.

Одним из атрибутов дескриптора <FORM> можно задать **имя формы**. Кроме этого, в форме может присутствовать атрибут **ACTION**, с помощью которого можно указывать программу-приложение, которая приступает к обработке введенных пользователем данных после того, как тот щелкнет на кнопке **Submit**. Приложение чаще всего находится на сервере и представляет собой программу, написанную на одном из языков программирования. Все тэги элементов управления располагаются между дескрипторами <FORM> и </FORM>. После щелчка на кнопке Submit, данные, введенные с помощью этих элементов (непосредственно в полях ввода или выбранные с помощью флажков, списков и других элементов управления), отсылаются на обработку приложению, указанному с помощью атрибута ACTION дескриптора <FORM>. В приложении разработчик предусматривает средства, с помощью которых результат, полученный после обработки полученных данных, отсылается пользователю.

Приведем пример очень простой формы (файл MyFirstForm.htm):

Листинг 2.6

```
<HTML>
<HEAD>
<title>Очень простая форма с атрибутом ACTION</title>
</HEAD>
<BODY bgcolor=f8f8ff >
     <FORM ACTION='./MyFirstApplication.exe'>
     Введите, пожалуйста, ваш адрес электронной почты: <P>
     <INPUT TYPE='text' NAME='email' VALUE='pvn@kmti.edu.ua' size='20'
     MAXLENGTH='25'><br>
     <INPUT TYPE='submit' name='submit' VALUE='Отравить'>
     </FORM>
</BODY>
</HTML>
```

В данном примере значение атрибута ACTION определяет, что при отправке данных они будут обработаны приложением MyFirstApplication.exe. После объявления формы идет строка текста. Затем следует управляющий элемент - поле ввода текстового типа, куда пользователь может вводить данные. Как можно видеть, она начинается с дескриптора <INPUT>, в котором имеется атрибут **TYPE**, определяющий тип поля ввода. В данном случае определено, что поле ввода должно быть типа строки текста (**text**). В дескрипторе <INPUT> могут определяться и другие

типы полей ввода. Эти типы будут рассмотрены далее в этом подразделе.

Все элементы формы могут иметь свое имя. Как уже упоминалось выше, после отправки данные формы передаются на обработку в программу-приложение. Это приложение должно проанализировать полученные данные и, в частности, определить, какое значение, какому элементу формы соответствует. Именно поэтому элементам формы присваиваются имена. Хотя, конечно, позволяется создать форму с безымянными элементами, и, при этом, все визуально будет выглядеть точно также. Однако, какая польза от формы, данные которой не смогут быть распознаны и, следовательно, обработаны получающим эту форму приложением?

В данном случае единственному элементу формы присвоено имя `email`. Значением по умолчанию для данного поля выбрана строка `pvn@kmti.edu.ua`. Размер поля ввода определяется с помощью атрибута `SIZE` дескриптора `<INPUT>`. В данном случае его значение равно 20-ти. Это означает, что длина видимой пользователю части поля ввода равна 20-ти символам. В поле ввода можно вводить больше 20-ти символов, но видимыми будут только 20. Чтобы увидеть оставшиеся символы следует воспользоваться клавишами управления курсором.

С помощью еще одного атрибута дескриптора `<INPUT>` - `MAXLENGTH` – можно определить максимальное количество символов, которое разрешается вводить в данное поле. В нашем примере значение этого атрибута равно 25-ти. Это означает, что браузер не разрешит пользователю ввести в данное поле ввода более 25 символов.

Следующая строка HTML-кода определяет кнопку подтверждения ввода. Кнопка подтверждения также определяется с помощью дескриптора `<INPUT>`. Однако в данном случае значение атрибута `TYPE` равно `Submit`. Помимо этого, здесь указано еще имя кнопки и значение надписи на кнопке.

Таким образом, дескриптор `<INPUT>` был использован для определения двух различных типов элементов интерфейса – поля ввода и кнопки подтверждения. При этом, тип элемента определяется с помощью атрибута `TYPE`.

Описания и примеры других типов элементов интерфейса (иначе – элементов управления), которые вы можете определять в дескрипторе `<INPUT>`, приведены ниже:

- `password` – элемент управления, используемый для создания поля ввода пароля. Текст, вводимый пользователем в таком поле, отображается в виде символов звёздочки « *».
  ```
  <INPUT> TYPE='password' NAME='myPassword' VALUE='' SIZE='10'>
  ```
- `button -` кнопка, обычно используемая для запуска программ на стороне клиента, которые мы будем писать при изучении программирования в третьем разделе конспекта.
  ```
  <INPUT> TYPE='button' NAME='myButton' VALUE='Щелкните на этой
  кнопке' onClick='myProg.js'>
  ```

- reset - кнопка, после щелчка на которой, все данные, введенные в форму, очищаются, и устанавливаются их значения по умолчанию.
  <INPUT> TYPE='reset' NAME='myResetButton' VALUE='Очистить'>
- image - аналог кнопки подтверждения Submit, с той лишь разницей, что вместо кнопки используется изображение. Эффект от щелчка на таком изображении полностью аналогичен эффекту от щелчка на кнопке Submit, т.е. после этого данные формы передаются на обработку приложению.
  <INPUT> TYPE='image' NAME='myImageSubmit' SRC='/submitImage.jpg'>
- radio – элемент управления, называемый «переключатель» или «радиокнопка». Набор состоит из нескольких переключателей, только один из которых может быть установлен в каждый отдельный момент. Выбрать установленный по умолчанию переключатель можно с помощью атрибута CHECKED дескриптора <INPUT>:
  <INPUT> TYPE='radio' NAME='myRadioButtons' VALUE='value1' CHECKED>
  <INPUT> TYPE='radio' NAME='myRadioButtons' VALUE='value2'>
  Все переключатели, имеющие одинаковое значение атрибута NAME, принадлежат к одной группе переключателей. В данном примере группа, названная myRadioButtons, состоит всего лишь из двух переключателей, первый из которых установлен по умолчанию. Если выбрать второй переключатель, первый автоматически сбрасывается. Когда данные такой формы отсылаются для обработки, то приложению передается только значение установленного переключателя. Таким образом, если после щелчка на кнопке Submit был установлен второй переключатель, приложение получит для обработки значение value2.
- checkbox - элемент управления, называемый «флажок». Набор флажков очень похож на набор переключателей с тем лишь только отличием, что одновременно здесь могут быть установлены сразу несколько (и даже все сразу) флажков. При отправке данных формы для обработки приложению передаются значения всех отмеченных флажков:
  <INPUT> TYPE='checkbox' NAME='myCheckboxButtons' VALUE='value1' CHECKED>
  <INPUT> TYPE='checkbox' NAME='myCheckboxButtons' VALUE='value2' CHECKED>
- hidden - используется для создания «скрытых» элементов, т.е. таких, которые не отображаются визуально, но присутствуют в форме. При отправке данных формы для обработки приложению передаются значения и всех «скрытых» элементов:
  <INPUT> TYPE='hidden' NAME='myHiddenField' VALUE='value1'>

<TEXTAREA>. С помощью текстового поля (text) тэга <INPUT>, описанного выше, пользователь может ввести лишь одну строку текста. Если же надо ввести не одну, а сразу несколько строк, то используется еще один очень полезный HTML-дескриптор - <TEXTAREA>. С

помощью этого дескриптора создается прямоугольная область нужного размера, в которой пользователю разрешается вводить текст. В качестве примера рассмотрим файл Text.Area.htm:

Листинг 2.7

```
<HTML>
<HEAD><title>Очень простая форма с  тэгом TEXTAREA</title></HEAD>
<BODY bgcolor=f8f8ff>
    <h2>Очень простая форма с  тэгом TEXTAREA</h2>
    <FORM>
          <TEXTAREA ROWS='5' COLS='30'>Это текст по умолчанию. Введите вместо
него то,  что вам требуется.
          </TEXTAREA><br>
     <INPUT TYPE='submit' name='submit' VALUE='Отправить'>
     </FORM>
</BODY>
</HTML>
```

Атрибут ROWS определяет количество отображаемых строк, т.е. фактически высоту прямоугольной области ввода. Атрибут COLS, в свою очередь, определяет количество столбцов, или, другими словами, количество символов, отображаемых в строке текста (длину области ввода). Для того, чтобы просмотреть текст, выходящий за рамки области, можно воспользоваться полосами прокрутки. Любой текст, находящийся между дескрипторами <TEXTAREA> и </TEXTAREA> , будет отображен в данной области, как текст по умолчанию.

<SELECT> - дескриптор для ввода в виде списка. Раскрывающийся (или выпадающий) список также является часто используемым элементом интерфейса. Назначение списка аналогично рассмотренным выше группам переключателей или флажков, однако он имеет совершенно иной внешний вид и, естественно иной синтаксис HTML. Список определяется дескриптором <SELECT>. С помощью атрибута Name можно определить имя списка. Помимо этого, важным моментом является определение способа выбора элементов списка. Это может быть либо единичный выбор (аналогично группе переключателей), либо множественный (аналогично группе флажков). По умолчанию разрешено выбирать только один элемент списка. Для того, чтобы разрешить множественный выбор, необходимо поместить в дескрипторе <SELECT> атрибут MULTIPLE.

В отличие от флажков и переключателей при использовании списка можно сделать видимыми всего лишь несколько вариантов выбора, а доступ к остальным можно предоставить с помощью полос прокрутки. С помощью атрибута SIZE дескриптора <SELECT> можно указать число вариантов выбора, которые можно оставить видимыми для пользователя. Если общее число вариантов выбора превышает значение атрибута SIZE, автоматически появляются полосы прокрутки, с помощью которых можно просмотреть остальные варианты.

Варианты выбора указываются с помощью дескрипторов <**OPTION**>. Их значения определяются сразу после этого дескриптора. Если же необходимо отправить обрабатывающему данные формы приложению какое-то другое значение, отличное от указанного после дескриптора

<OPTION>, его можно определить с помощью атрибута VALUE этого дескриптора. Рассмотрим дескрипторы <SELECT> на примере следующего файла:

Листинг 2.8

```
<HTML>
     <HEAD><title>Очень простая форма с  дескриптором
         SELECT</title> </HEAD>
<BODY>
     <h2>Очень простая форма с дескриптором SELECT.</h2>
     <FORM>
          <SELECT NAME='mySelectList' MULTIPLE SIZE='3'>
            <OPTION value='1'>Пеличева Оксана
            <OPTION value='2'>Крихтина Ирина
            <OPTION value='3'>Суховая Екатерина
            <OPTION value='4'>Шахназарян Мелинэ
            <OPTION value='5'>Чернец Юлия
          </SELECT><br>
        <INPUT TYPE='submit' name='submit' VALUE='Отравить'>
     </FORM>
</BODY>
</HTML>
```

В коде приведенного выше примере можно видеть, что в нем создан список с именем mySelectList и пятью вариантами выбора. С помощью атрибута MULTIPLE пользователю разрешается выбирать сразу несколько вариантов. Поскольку число одновременно отображаемых вариантов ограничено тремя (атрибут SIZE=3) пользователю придется использовать полосы прокрутки, чтобы просмотреть два остальных варианта. При выборе элемента (элементов) списка и отправке данных формы для обработки, приложению будут переданы не те значения, которые были указаны после дескриптора <OPTION>, а те, которые были указаны с помощью атрибута VALUE. Например, при выборе вариантов *Пеличева Оксана* и *Шахназарян Мелинэ* приложению будут отправлены значения *1* и *4*.

Для того, чтобы получше изучить все типы элементов интерфейса, определяемые с помощью дескрипторов <INPUT>, <TEXTAREA> и <SELECT>, рекомендуется поэкспериментировать с ними, создав HTML-файлы в виде выше приведенных примеров. В результате этих экспериментов вы должны уметь определять, в каком случае необходимо использовать тот или иной тип интерфейса.

Обратим внимание на еще один момент – каким образом данные из формы попадают в программу на языке JavaScript? Ранее, до появления JavaScript, данные формы обрабатывались единственным образом - отправлялись на удаленный сервер. Как отмечено выше для этой цели тэг <FORM> содержит параметр ACTION, определяющий URL приложения на сервере, по которому следовало направить данные формы. Пользователь нажимает на кнопку SUBMIT, и данные из формы передаются по указанному URL, где должны присутствовать

программы–приложения для обработки поступающих данных. Этот метод подробно рассматривается в другом конспекте лекций [5], посвященном основам программирования на стороне сервера.

С появлением `JavaScript` появилась возможность обрабатывать большую часть данных формы на локальной машине, то есть на стороне клиента, отправляя для обработки на сервер лишь то, что невозможно (или нецелесообразно) обрабатывать на стороне клиента. Данные из формы попадают в программу на языке `JavaScript` с помощью событий, встроенных в язык HTML. Виды событий и приемы их обработки с помощью функций на языке `JavaScript` подробно рассмотрены в следующем разделе настоящего конспекта.

Следует также знать, что в HTML5 (новой версии HTML) введены дополнительные тэги и атрибуты, позволяющие во многих случаях (проверка корректности ввода данных полей и форм, правила перехода по полям и др.) обойтись без использования программирования [10, http://htmlbook.ru/samhtml5/formy].

Для закрепления материала необходимо перейти в подраздел 2.8 и выполнить упражнение 2.7.

2.8. Упражнения

Упражнение 2.1.

Цель упражнения: усвоить принципы создания электронных документов средствами HTML с использованием листов стилей (CSS).

Конечный результат - вариант простейшего сайта, состоящего из трёх страниц, с использованием присоединяемых листов стилей.

Этапы выполнения упражнения (6 этапов).

Этап 1. Повторите пример, приведённый в подразделе 2.2 в листинге 2.1. Создайте файл со стилями, приведёнными в примере и сохраните его под именем Petrov.css (вместо Petrov вы должны подставить свою фамилию).

Этап 2. Присоедините файл Petrov.css с помощью тэга <link .../> ко всем трём веб-документам (страницам), созданным при выполнении упражнения 1. Проведите эксперименты, меняя значения каждого свойства стиля в файле Petrov.css, чтобы убедиться, что все задаваемые вами стили работают, и работают они на каждой из трёх страниц. Цвета подбирайте с помощью RGB-палитры, размещённой на сайте RosDesign.com: http://rosdesign.com/design/politraofdesign.htm

.

Этап 3. Отредактируйте файл со стилями, дополнив его правилом стилей для тэга <ol>..., - задайте для этого тэга размер шрифта, как 14px.

Этап 4. Создайте ещё один файл с именем exercise_2_1.html в папке exercises и поместите в нём ответы на первые 10 контрольных вопроса из подраздела 2.9 (ответы — своими словами и не более трёх строк). Озаглавьте эту страницу так: «Упражнение 2.1. Ответы на контрольные вопросы по подразделам 2.1 и 2.2». Вопросы-ответы пронумеруйте также с помощью тэга <ol>..., как это вы делали в файле exercise_1.html.

На странице должны быть ссылки для возврата на домашнюю страницу и страницу со списком упражнений. Все остальные заголовки и тексты пусть будут такими же, что и на странице с упражнением 1.

Этап 5. Дополните страницу со списком упражнений ссылкой для перехода к странице, созданной на этапе 4. На странице должен появиться отображаемый текст ссылки: «Упражнение 2.1». Проверьте работоспособность ссылки.

Этап 6. Присоедините файл со стилями к странице созданной на этапе 4. Убедитесь, что задаваемые Вами стили работают и на этой странице.

Упражнение 2.2.

Цель упражнения: усвоить принципы использования классов и идентификаторов стилей на примере тэга ... и CSS- стилей блочной модели.

Конечный результат - вариант простейшего сайта, состоящего из трёх страниц, с использованием тэга ... и CSS- стилей блочной модели.

Для выполнения упражнения используйте справочные пособия по CSS-стилям, приведённые в конце этого подраздела.

Этапы выполнения упражнения (4 этапа).

Этап 1. Повторите пример, приведённый в листинге 2.3 подраздела 2.2 с присоединением стилей из листинга 2.4, преобразив свою домашнюю страницу так, чтобы в браузере она имела вид, подобный тому, который показан на рис. 2.2.

Этап 2. Присоедините css-файл с помощью тэга <link .../> ко всем документам своего персонального сайта.

Этап 3. Создайте ещё один файл с именем exercise_2_2.html и поместите в этот файл ответы на 13 контрольных вопросов (с 11 по 23-й вопрос).

Этап 4. По аналогии с предыдущем упражнением, поместите в файл exercises.html ссылку на файл exercise_2_2.html. В файле exercise_2_2.html должны быть те же самые ссылки, что и в

exercise_2_1.html (для перехода к домашней странице и к странице со списком упражнений). Проверьте в браузере работоспособность каждой вновь созданной ссылки.

Примеры того, что у вас должно быть создано, можно посмотреть среди работ студентов на сайте по адресу: http://pvn.ho.ua/.

Упражнение 2.3.

Цель упражнения: усвоить принципы использования листов стилей для создания гиперссылок.

Конечный результат - создание горизонтального меню ссылок в виде блока для использования на страницах учебного сайта.

Этапы выполнения упражнения.

Этап 1. Повторите создание блока с горизонтальным меню для своего сайта, воспользовавшись примером того, что сделал студент Дудка Никита на своём персональном сайте: http://pvn.ho.ua/pvn.org.ua/www/students/10kmk183/www/index.html.

Этап 2. Вставьте блок с меню в блок с идентификатором content (id="content") так, чтобы меню отображалось под заголовком (под блоком header) на каждой странице. Проверьте работоспособность ссылок и, если ссылки не работают, то сделайте необходимые исправления.

Этап 3. Создайте ещё один файл с именем exercise_2_3.html и поместите в этот файл ответы на 7 контрольных вопросов (с 24-го по 30-й вопрос).

Этап 4. По аналогии с предыдущем упражнением, поместите в файл exercises.html ссылку на файл exercise_2_3.html. В файле exercise_2_3.html должны быть те же самые ссылки, что и в exercise_2_1.html, exercise_2_2.html (для перехода к домашней странице и к странице со списком упражнений). Проверьте в браузере работоспособность каждой вновь созданной ссылки.

Упражнение 2.4.

Цель упражнения: усвоить принципы использования листов стилей для управления изображениями.

Конечный результат - создание страницы "О себе" в составе своего учебного сайта.

Этапы выполнения упражнения.

Этап 1. Разработайте страницу "О себе" для своего сайта, воспользовавшись примером того, что сделал студент Дудка Никита на своём персональном сайте: http://pvn.ho.ua/pvn.org.ua/www/students/10kmk183/www/index.html. На странице должно быть не менее четырёх абзацев с принадлежащими лично вам фото и информацией. В абзацах должна быть информация: о себе, о школе, почему выбрали КГМТУ и специальность "Водные биоресурсы",

интересы по выбранной специальности и в свободное время, хобби. В каждом абзаце должно быть размещено не менее одной фотографии. При этом необходимо выполнить условия, при которых:

- фотографии должны размещаться то по левой, то по правой стороне абзаца с выравниванием по соответствующей стороне абзаца;
- не должны сливаться с текстом следующего или предыдущего абзаца;
- обеспечивались необходимые отступы между текстом и изображением;
- рамка выглядела так, как это Вам хотелось бы.

Кроме того, на этом примере нужно научиться управлять фоновыми изображениями: для страницы в целом и для блоков внутри страницы.

Этап 2. Отредактируйте, если это необходимо, стили страницы в соответствии с шаблоном: блоки "header", "content", "footer", "menu" в блоке "content". Проверьте работоспособность ссылок и, если ссылки не работают, то сделайте необходимые исправления.

Этап 3. Создайте ещё один файл с именем exercise_2_4.html и поместите в этот файл ответы на 7 контрольных вопросов (с 31-го по 37-й вопрос).

Этап 4. По аналогии с предыдущем упражнением, поместите в файл exercises.html ссылку на файл exercise_2_4.html. В файле exercise_2_4.html должны быть те же самые ссылки, что и в exercise_2_1.html, exercise_2_2.html, exercise_2_3.html (для перехода к домашней странице и к странице со списком упражнений). Проверьте в браузере работоспособность каждой вновь созданной ссылки.

Упражнение 2.5.

Цель упражнения: усвоить принципы создания таблиц и использования листов стилей для управления дизайном таблиц.

Конечный результат - редактирование страницы "Упражнения" в составе своего учебного сайта, с использованием табличного дизайна.

Этапы выполнения упражнения.

Этап 1. Отредактируйте страницу "Упражнения" своего сайта, воспользовавшись примером того, что сделал студент Петров на своём персональном сайте: http://pvn.ho.ua/pvn.org.ua/www/students/10kmk000/www/exercises/exercises.html (смотрите, также примеры других студентов, бывших первокурсников, имеющих свои персональные сайты на pvn.ho.ua. После выполнения этого упражнения страница "Упражнения" на вашем сайте должна выглядеть также, но, конечно, с вашим неповторимым стилевым оформлением. Для оформления ссылок в первом столбце таблицы, вспомните предыдущий подраздел и воспользуйтесь псевдоклассами стилей для создания динамических эффектов с помощью псевдоклассов.

Этап 2. Отредактируйте, если это необходимо, стили страницы в соответствии с шаблоном: блоки "header", "content", "footer", "menu" в блоке "content". Проверьте работоспособность ссылок блока "menu", и, если ссылки не работают, то сделайте необходимые исправления.

Этап 3. Создайте ещё один файл с именем exercise_2_5.html и поместите в этот файл ответы на 7 контрольных вопросов (с 38-го по 44-й вопрос).

Этап 4. По аналогии с предыдущем упражнением, поместите в файл exercises.html ссылку на файл exercise_2_5.html. В файле exercise_2_5.html должны быть те же самые ссылки, что и в exercise_2_1.html, exercise_2_2.html, exercise_2_3.html, exercise_2_4.html (для перехода к домашней странице и к странице со списком упражнений). Проверьте в браузере работоспособность каждой вновь созданной ссылки.

Упражнение 2.6.

Цель упражнения: усвоить принципы использования слоёв.

Конечный результат - создание страницы "Упражнение 2.6" (exercise_2_6.html) с использованием слоёв и добавление её в состав своего персонального сайта.

Этапы выполнения упражнения.

Этап 1. Повторите последний пример, приведённый в подразделе 2.6, создав ещё одну страницу своего сайта и сохранив её в файле exercise_2_6.html. Отредактируйте этот пример так, чтобы при щелчке мыщью слои менялись местами - нижний слой становился верхним, а верхний перемещался вниз. Для появления такого динамического эффекта вспомните подраздел 2.4 и воспользуйтесь псевдоклассами стилей (посказка: псевдокласс :hover можно использовать не только для гиперссылок, но и для других тэгов, в том числе - для тэга ... .

Этап 2. Отредактируйте, если это необходимо, стили страницы в соответствии с шаблоном: блоки "header", "content", "footer", "menu" в блоке "content". Проверьте работоспособность ссылок блока "menu", и, если ссылки не работают, то сделайте необходимые исправления.

Этап 3. Поместите на эту же страницу ответы на четыре контрольных вопроса (с 45-го по 48-й вопрос).

Этап 4. По аналогии с предыдущем упражнением, поместите в файл exercises.html ссылку на файл exercise_2_6.html. В файле exercise_2_6.html должны быть те же самые ссылки, что и в exercise_2_1.html, exercise_2_2.html, exercise_2_3.html, exercise_2_4.html, exercise_2_5.html (для перехода к домашней странице и к странице со списком упражнений). Проверьте в браузере работоспособность каждой вновь созданной ссылки.

Упражнение 2.7.

Цель упражнения: усвоить принципы создания форм.

Конечный результат - создание страницы "Упражнение 2.7" (exercise_2_7.html) с использованием форм и добавление её в состав своего персонального сайта.

Этапы выполнения упражнения.

Этап 1. Повторите последний пример, приведённый в подразделе 2.7, посвящённом работе с формами, создав ещё одну страницу своего сайта и сохранив её в файле exercise_2_7.html. Отредактируйте этот пример так, чтобы список содержал фамилии и имена студентов вашей группы.

Этап 2. Отредактируйте, если это необходимо, стили страницы в соответствии с шаблоном: блоки "header", "content", "footer", "menu" в блоке "content". Проверьте работоспособность ссылок блока "menu", и, если ссылки не работают, то сделайте необходимые исправления.

Этап 3. Поместите на эту же страницу ответы на восемь контрольных вопросов (с 49-го по 56-й).

Этап 4. По аналогии с предыдущем упражнением, поместите в файл exercises.html ссылку на файл exercise_2_7.html. В файле exercise_2_7.html должны быть те же самые ссылки, что и в exercise_2_1.html, exercise_2_2.html, exercise_2_3.html, exercise_2_4.html, exercise_2_5.html, exercise_2_6.html (для перехода к домашней странице и к странице со списком упражнений). Проверьте в браузере работоспособность каждой вновь созданной ссылки.

2.9. Контрольные вопросы

1. Что такое HTML ?
2. Что такое тэг или дескриптор?
3. Приведите примеры тэгов контейнерного типа и одиночных тэгов.
4. Какой минимальный набор тэгов должен присутствовать в HTML-документе?
5. Что такое свойство или атрибут тэга?
6. В каком тэге — открывающем или закрывающем можно указывать атрибуты (свойства) для этого тэга?
7. Какой разделитель используется для отделения одного свойства (атрибута) от другого и от наименования тэга?
8. Что такое CSS ?
9. С помощью какого дескриптора присоединяются внешние (или присоединяемые) стили?
10. Ставятся ли угловые скобки вокруг наименования тэга при написании правила стилей для этого тэга?
11. Какие свойства стилей предназначены для задания высоты и ширины HTML-блока?

12. Какие значения высоты и ширины будет иметь HTML-блок, если не будут указаны высота и ширина HTML-блока?
13. Перечислите все стили, которые можно использовать для задания размера внешнего отступа вокруг HTML-блока.
14. Перечислите все стили, которые можно использовать для задания размера отступа вокруг содержимого HTML-блока.
15. Какие свойства стиля необходимо задавать обязательно, чтобы отображалась рамка вокруг HTML-блока?
16. Как задаются классы стилей?
17. В чём преимущество задания классов стилей?
18. Как задаётся идентификатор стиля?
19. В чём различие классов стилей и идентификаторов стилей?
20. Зачем предназначен тэг ... ?
21. В чём различие частных, внутренних и внешних (присоединяемых) листов стилей?
22. С помощью какого свойства (атрибута) описываются частные стили?
23. С помощью какого дескриптора описываются внутренние стили?
24. Что такое гиперссылка и какой дескриптор служит для создания гиперссылки?
25. Что такое абсолютная ссылка?
26. Что такое относительная ссылка?
27. Как создаётся гиперссылка на какое-либо место в одном и том же документе?
28. Приведите пример создания ссылки для перехода к элементу в одном и том же документе, используя атрибут ID.
29. Что такое псевдокласс стиля применительно к гиперссылке?
30. Приведите пример псевдокласса для гиперссылки, создающего эффект изменения цвета при наведении указателя мыши над элементом привязки.
31. Приведите пример тэга для вставки изображений с атрибутами src, alt и title.
32. Для чего предназначен атрибут alt в тэге для вставки изображений?
33. Для чего предназначен атрибут title в тэге для вставки изображений?
34. Как можно задать размеры изображения с помощью атрибута style?
35. Как можно задать размещение изображения относительно текста по горизонтали с помощью атрибута style и стиля float?
36. Назначение стиля clear и пример его использования с помощью атрибута style?
37. Когда и как используются стили: "clear: all"; "clear:left"; "clear:right"?
38. Приведите пример создания html-таблицы с двумя строками и тремя солбцами.
39. Приведите пример задания рамки таблицы толщиной 2px с помощью атрибута style.

40. Приведите пример задания рамки ячейки толщиной 1px с помощью атрибута style.
41. Приведите пример задания ширины столбцов таблицы размером 25% с помощью атрибута style.
42. Приведите пример задания высоты строк таблицы размером 30px с помощью атрибута style.
43. Приведите пример задания фонового изображения для таблицы с помощью атрибута style.
44. Приведите пример задания меняющегося фонового изображения для таблицы с помощью атрибута style, используя псевдокласс :hover.
45. Понятие слоя в HTML?
46. С помощью каких правил стилей создаются слои в HTML?
47. Приведите пример двух слоёв, частично перекрывающих друг друга?
48. Какое значение стиля z-index необходимо задать, чтобы описываемый этим стилем слой перекрывал все остальные слои при наложении?
49. Для чего предназначен контейнер <FORM> и </FORM>?
50. Что такое элементы управления формы?
51. Для чего предназначен атрибут ACTION тэга <FORM>?
52. Для чего предназначен атрибут NAME дескриптора <FORM>?
53. С помощью какого атрибута, кроме тэга NAME, можно именовать контейнер <FORM> и </FORM>?
54. Перечислите все типы элементов интерфейса (элементов управления), которые вы можете определять с помощью тэга <INPUT>.
55. Для ввода каких данных служит контейнер <TEXTAREA> и </TEXTAREA>?
56. Для ввода каких данных служит контейнер <SELECT> и </SELECT>?

2.10. Литература и веб-источники к разделу

[1] Браун Марк, Хоникатт Джерри и др. Использование HTML 4.

[2] Дронов В.А. Javascript в Web-дизайне. – СПб.: БХВ-Петербург, 2004. - 880 с.

[6] http://pvn.ho.ua/pvn.org.ua/www/#2 - лекции на сайте преподавателя (лекция по HTML+CSS)

[7] http://docs.google.com/ – бесплатный сервис для хранения файлов

[8] http://analog.com.ua - HTML - справочник (наиболее простой)

[9] http://css.manual.ru/ - CSS-справочник (наиболее простой)

[10] http://htmlbook.ru/ - HTML, CSS, вёрстка веб-страниц (справочники, руководства, примеры - всё по разработке веб-документов)

[11] http://www.intuit.ru/department/internet/css/1/ - лекции по таблицам стилей (CSS)

[12] http://www.intuit.ru/department/internet/css/ - курс обучения использованию CSS

[13] http://rosdesign.com/design/politraofdesign.htm - RGB-цвета безопасной палитры

[14] http://www.intuit.ru/department/internet/html5/ - Введение в HTML5

Резюме. Если вы получили положительную итоговую оценку за этот раздел, то вы стали обладателем средств для самостоятельной разработки электронных HTML-документов. Вы понимаете, как самостоятельно преобразовать в собственных целях дизайн понравившихся вам веб-страниц, бесчисленное количество которых имеется в Сети.

Но, главное для изучаемой дисциплины, вы получили необходимый минимум знаний по созданию электронных документов, как среды, в которой размещаются и выполняются программы, поскольку для вас основная задача — освоение программирования, которому посвящены все последующие разделы настоящего конспекта лекций.