МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА
И ПРОДОВОЛЬСТВИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ОБРАЗОВАНИЯ, НАУКИ И КАДРОВ

Учреждение образования
«БЕЛОРУССКАЯ ГОСУДАРСТВЕННАЯ
СЕЛЬСКОХОЗЯЙСТВЕННАЯ АКАДЕМИЯ»

И. В. Шараева, Т. С. Прокопова, В. Г. Ракутин

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

ОСНОВЫ ВЕБ-ПРОЕКТИРОВАНИЯ

Рекомендовано учебно-методическим объединением по образованию в области сельского хозяйства в качестве курса лекций для студентов учреждений высшего образования, обучающихся по специальности 1-74 06 04 Техническое обеспечение мелиоративных и водохозяйственных работ

Горки
БГСХА
2017

УДК 004.9(075.8)
ББК 32.81я73
Ш25

Рекомендовано методической комиссией факультета механизации сельского хозяйства 30.05.2016 (протокол № 9) и Научно-методическим советом БГСХА 01.06.2016 (протокол № 9)

Авторы:
старший преподаватель *И. В. Шараева*;
ассистент *Т. С. Прокопова*;
кандидат экономических наук, доцент *В. Г. Ракутин*

Рецензенты:
доктор экономических наук, доцент *А. Г. Ефименко*;
кандидат экономических наук, доцент *А. Б. Гедранович*;
кандидат технических наук, доцент *А. В. Пашкевич*

Шараева, И. В.

Ш25 Информационные технологии. Основы веб-проектирования : курс лекций / И. В. Шараева, Т. С. Прокопова, В. Г. Ракутин. – Горки : БГСХА, 2017. – 58 с.

ISBN 978-985-467-705-7.

Рассмотрены основные особенности языка HTML и каскадных таблиц стилей (CSS).

Для студентов учреждений высшего образования, обучающихся по специальности 1-74 06 04 Техническое обеспечение мелиоративных и водохозяйственных работ.

УДК 004.9(075.8)
ББК 32.81я73

ISBN 978-985-467-705-7

ВВЕДЕНИЕ

В связи с интенсивным развитием веб-технологий и возможностями их использования в различных областях деятельности владение элементарными навыками создания веб-ресурсов становится частью информационной культуры современного специалиста.

Основной объем информации, доступной в сети Интернет, размещается во Всемирной паутине (World Wide Web (WWW)) – информационной системе, подобной гигантской библиотеке. В этой библиотеке информация представлена в виде связанных между собой веб-страниц. Собрание страниц, объединенных некоторой общей темой, образует веб-сайт. Посмотреть любую веб-страницу можно с помощью специальных программ – браузеров, наиболее популярными из которых являются Opera, Mozilla, Google Chrome.

Разработку веб-сайта, однако, можно проводить и на компьютере, не имеющем выхода в Интернет. Для создания веб-сайта могут быть использованы различные инструментальные средства веб-редактирования, прежде всего специальные редактирующие программы и программные пакеты, реализующие принципы визуального редактирования веб-страниц (Microsoft FrontPage, Adobe Dreamweaver и пр.). Кроме того, современные версии (2007, 2010, 2013) офисного пакета Microsoft Office и ряд других прикладных программ дают возможность сохранять обрабатываемые в них документы в веб-совместимом формате, позволяющем размещать их на сайте и просматривать в сети Интернет. Однако для понимания принципов веб-проектирования студентам необходимо знание языка гипертекстовой разметки HTML, как основы веб-представления текстовых и медиаматериалов.

Каскадные таблицы стилей (CSS) открывают совершенно новые возможности в веб-проектировании. Сделать качественный дизайн веб-сайта без использования CSS очень проблематично.

С помощью данного курса лекций студенты знакомятся с основами языка HTML и возможностями использования каскадных таблиц стилей для оформления веб-страниц.

Методическая разработка состоит из трех лекций: введение в веб-проектирование, создание веб-страниц средствами языка HTML и оформление веб-страниц средствами CSS, а также из двух приложений, в которых приведены основные теги HTML и свойства CSS.

Лекция 1. ВВЕДЕНИЕ В ВЕБ-ПРОЕКТИРОВАНИЕ

Цель: изучить этапы создания веб-сайтов, ознакомиться с основными элементами веб-страниц и особенностями их создания, получить представление о средствах реализации динамических веб-страниц.

1.1.Основные определения.
1.2. Логическая структура веб-сайта.
1.3. Веб-страница с точки зрения файловой системы. Физическая структура веб-сайта.
1.4. Основные элементы веб-страницы: заголовок, текст, графические изображения, гиперссылки, таблицы, формы.
1.5. Средства создания веб-страниц. Возможности редакторов визуального проектирования. Просмотр веб-страниц в браузерах.
1.6. Методы реализации динамических веб-страниц.
1.7. Этапы создания веб-сайта.

1.1. Основные определения

Введем основные определения.

WWW (World Wide Web – Всемирная паутина) – совокупность взаимосвязанных гипермедийных документов.

Веб-страница – документ WWW, содержащий форматированный текст, мультимедийные объекты, гиперссылки, активные компоненты.

Веб-сайт (далее – сайт) – группа веб-страниц, связанных единой темой, схемой оформления и гипертекстовыми ссылками.

HTML (HyperText Markup Language – язык гипертекстовой разметки) – язык создания веб-страниц.

Веб-сервер – программа, позволяющая хранить и пересылать веб-страницы, например, Apache, IIS (Internet Information Services) и др.

Веб-браузер (далее – браузер) – программа, предназначенная для просмотра веб-страниц, например, Internet Explorer, Google Chrome, Mozilla Firefox и др. Браузер по заданному URL доставляет веб-страницу на компьютер пользователя и интерпретирует код HTML.

1.2. Логическая структура веб-сайта

Подготовка к созданию веб-сайта. Первым этапом в создании веб-сайта является сбор информации. Прежде всего необходимо определить цель создания сайта, задачи, которые планируется решить по-

средством сайта, определить потенциальную аудиторию – от всего этого зависит содержание и внешний вид будущего сайта. Следует продумать тексты, которые будут размещены на сайте. Также нужно подготовить в электронном виде мультимедийные объекты (графику, анимацию, звук), которые будут участвовать в формировании сайта.

Следующий этап в формировании сайта – проектирование его *логической структуры*. На этом этапе необходимо разбить материал на отдельные логические разделы (в дальнейшем каждый раздел будет представлять собой отдельную страницу) и продумать связи между ними. Любой сайт имеет домашнюю страницу – страницу, с которой начинается просмотр сайта, – включающую общую информацию о содержании сайта, ссылки на остальные страницы сайта (или на основные его разделы, если страниц слишком много). Полезно предварительно на листе бумаги изобразить карту сайта – графическое представление логической структуры сайта, где прямоугольниками обозначены страницы, а линиями – связи между ними (рис. 1).

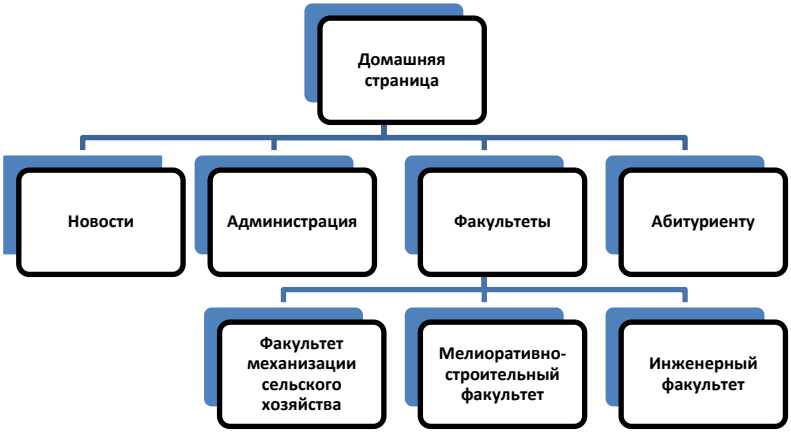

Рис. 1. Карта сайта учреждения высшего образования

1.3. Веб-страница с точки зрения файловой системы. Физическая структура сайта

Как правило, веб-страница представляет собой целый набор файлов. Прежде всего, каждая веб-страница – это отдельный HTML-документ. С точки зрения файловой системы это файл с расширением

.htm или **.html**. Следует помнить, что мультимедийные объекты не внедряются непосредственно в HTML-документ. В HTML-документе есть лишь специальное указание на то, что в данной области страницы должен появиться, например, рисунок, и указание пути к нему, а сам рисунок хранится как отдельный файл. То же касается и других мультимедийных объектов. Если же на странице предполагается наличие интерактивности (т. е. внешний вид страницы может зависеть от действий просматривающего ее пользователя), это реализуется средствами специальных языков программирования, используемых для создания активных компонентов, которые также могут храниться как отдельные файлы, прикрепленные к веб-странице.

Таким образом, количество файлов, участвующих в образовании сайта, как правило, гораздо больше количества страниц. Поэтому прежде чем приступать к созданию каждой страницы, необходимо позаботиться о размещении файлов сайта, т. е. определить физическую структуру сайта – способ размещения файлов по папкам. При этом желательно соблюдать следующие правила:

1. Для хранения файлов сайта следует создать отдельную папку (так называемую корневую папку сайта). Это необходимо сделать в связи с тем, что сайт создается на локальном компьютере (который может и не иметь выхода в Интернет). Однако после того, как сайт будет готов, он должен пройти процедуру публикации. Только после этого сайт станет доступен всем пользователям WWW. В общих чертах публикация сводится к копированию всех документов сайта в один из подкаталогов сервера. Таким образом, если с самого начала хранить все документы сайта в отдельной папке, для публикации достаточно будет скопировать на сервер содержимое корневой папки.

2. Внутри корневой папки сайта можно создать любую систему подкаталогов для лучшей систематизации хранения документов (рис. 2). В предложенной схеме файлы раскладываются по папкам в соответствии с их типом. Может быть выбран и другой алгоритм распределения файлов по папкам – например, в соответствии с тематикой.

3. HTML-документ, задающий домашнюю страницу сайта, должен носить имя **index.htm** и храниться непосредственно в корневой папке сайта (а не в одном из ее подкаталогов). При выполнении этого правила адрес вашего сайта после публикации будет выглядеть примерно так: **http://адрес_сервера/каталог_сервера** (имя файла не указывается!). Если же назвать домашнюю страницу по-другому или разместить ее в некоем подкаталоге корневой папки, в адресе придется указывать

название подкаталога и имя файла, например:

http://адрес_сервера/каталог_сервера/html/glavnaja.htm.

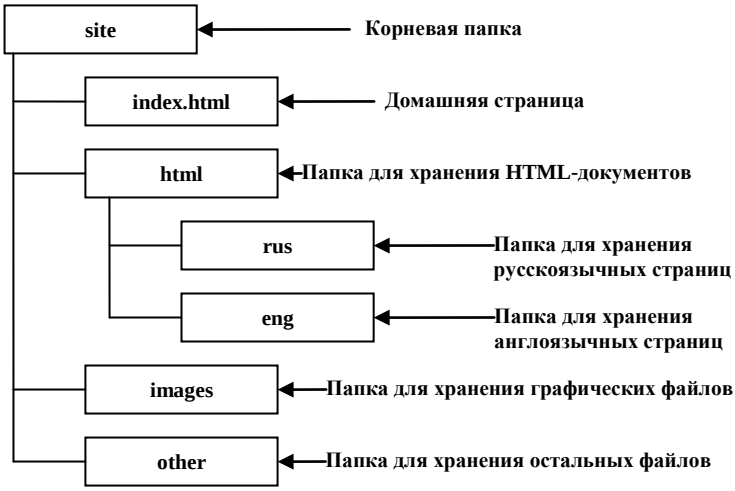

Рис. 2. Физическая структура сайта

Таким образом, невыполнение данного правила существенно удлинит адрес вашего сайта. Все остальные документы можно распределять по папкам произвольным образом.

4. В именах всех файлов и папок сайта следует использовать только строчные латинские буквы и (или) цифры. Это связано с тем, что многие платформы, на которых строятся серверы, не поддерживают буквы национальных алфавитов (в том числе русского) в именах файлов и папок и различают регистр. При невыполнении данного правила после публикации могут возникнуть проблемы с чтением документов сайта.

После подготовки физической структуры сайта можно приступать к формированию каждой страницы в отдельности. Далее остановимся на наиболее часто встречающихся элементах страниц.

1.4. Основные элементы веб-страницы

Заголовок веб-страницы. Заголовок веб-страницы представляет собой текст, отображающийся в строке заголовка окна браузера при просмотре страницы. Заголовок веб-страницы должен содержать несколько слов (на любом языке), которые наиболее полно и точно характеризуют содержимое страницы. Это связано, во-первых, с тем, что

заголовок первым появляется при загрузке. Поскольку скорость загрузки страниц может быть недостаточно высокой, возможно, пользователь некоторое время будет видеть лишь текст строки заголовка. Во-вторых, при создании закладок на страницы в качестве имени закладки по умолчанию предлагается именно заголовок страницы, и это тоже аргумент в пользу того, что к выбору заголовка нужно подходить серьезно. Наконец, текст строки заголовка имеет больший вес при индексации поисковыми системами, поэтому заголовок должен содержать как можно больше ключевых слов.

Текст веб-страницы. Как правило, наибольшую смысловую нагрузку на странице несет текст. Следует помнить, что страница, скорее всего, будет просматриваться пользователем с монитора (а не в печатном виде), и нужно сделать текст удобным для чтения. Текст не должен быть слишком длинным (за исключением тех случаев, когда это действительно необходимо). Форматировать его на веб-страницах можно примерно так же, как в обычных текстовых процессорах: указывать гарнитуру текста, способ начертания, цвет текста, его размер, интервал между строками, поля в тексте, цвет фона и т. д.

Для удобства чтения следует соблюдать некоторые правила. Более удобным для чтения считается текст, имеющий большие поля. Интервал между строками должен быть тем больше, чем меньше размер шрифта. Цвета текста и фона должны быть достаточно контрастными. Не следует использовать слишком яркие цвета для оформления веб-страницы.

Особое внимание следует уделить выбору гарнитуры. На странице должно быть использовано не более двух различных гарнитур шрифта. Исключение может составлять лишь случай, когда особой гарнитурой оформляются текстовые заголовки. Однако следует помнить, что текст будет отображаться в указанной вами гарнитуре лишь в том случае, если соответствующая гарнитура установлена на компьютере пользователя, просматривающего страницу. В противном случае соответствующий фрагмент текста отобразится в гарнитуре, заданной для браузера пользователя в качестве гарнитуры по умолчанию.

Графические изображения. Теперь рассмотрим основные правила вставки изображений на веб-страницы. Графические изображения (как и другие мультимедийные объекты) хранятся в отдельных файлах, а в HTML-документе присутствуют лишь адреса этих файлов. Для того чтобы правильно указать источник изображения, необходимо различать ***абсолютные*** и ***относительные*** адреса файлов.

Пусть необходимый нам рисунок хранится на диске **D:** в папке **Мои рисунки** и соответствующий файл носит имя **ris.gif**. Если в каче-

стве пути к файлу записать **D:/Мои рисунки/ris.gif** (абсолютный путь), то при просмотре страницы браузер посетителя будет искать указанный файл на его же компьютере. Очевидно, что по указанному адресу нужный файл обнаружен не будет и вместо рисунка на странице отобразится пустая рамка. Поэтому все необходимые графические файлы должны храниться в корневой папке сайта или в одном из ее подкаталогов. Далее адрес графического файла следует указать в относительном виде: относительно того HTML-документа, в который требуется вставить соответствующее изображение.

Пусть нам необходимо вставить в HTML-документ **page.htm**, который находится по адресу **site/html/rus**, изображение из файла **site/images/ris.gif** (рис. 3). Относительный путь указывает, как от HTML-документа можно «добраться» до графического файла. В нашем случае переходим в надкаталог: попадаем из папки **rus** в папку **html**. Еще раз переходим в надкаталог и попадаем в папку **site**. В папке **site** открываем папку **images** и находим там нужный файл. Значит, относительный адрес следует записать так: **../../images/ris.gif**. Две точки здесь символизируют переход в надкаталог. Если изображение из того же файла **ris.gif** необходимо вставить в документ **index.htm**, хранящийся в папке **site**, то относительный адрес следует записать так: **images/ris.gif**, т. е. в папке, где находится HTML-документ (**site**), нужно открыть папку **images** и выбрать графический файл.

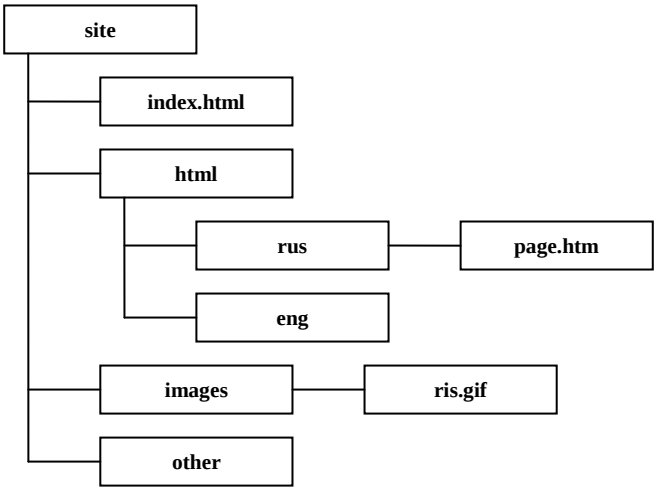

Рис. 3. Путь к графическому файлу

Опишем некоторые параметры изображения:

Размер. Размер указывают в пикселах или в процентах от ширины родительского элемента – того, в который помещают изображение (окно браузера, ячейка таблицы и др.). При указании размера изображения следует помнить о том, что разрешение монитора пользователя может отличаться от разрешения вашего монитора. В таком случае при указании размера изображения в пикселах компоновка страницы может существенно нарушиться, тогда как при указании размера в процентах рисунок будет занимать указанную часть монитора независимо от разрешения.

Альтернативный текст. Альтернативный текст отображается вместо рисунка в режиме отключения графики или неграфическими браузерами. Многие браузеры отображают альтернативный текст в качестве всплывающей подсказки при наведении указателя мыши на изображение. Поисковые системы могут производить поиск изображений на веб-страницах, опираясь на название соответствующего графического файла или на альтернативный текст.

Графика на веб-страницах может использоваться и в качестве фоновых изображений (всевозможные текстуры или даже фотографии как фон страниц). Здесь следует помнить о том, что фоновое изображение не будет отображаться, если отключить загрузку графики.

Гиперссылки. Одним из важнейших элементов любой веб-страницы являются гиперссылки. При создании гиперссылки разработчик должен задать указатель гиперссылки – элемент страницы, который станет ссылкой, а также адресную часть ссылки – адрес ресурса, на который указывает гиперссылка.

Указателем ссылки, как правило, является фрагмент текста или графическое изображение. Очень часто текст, являющийся ссылкой, подчеркнут и выделен цветом. Что касается графики, то на многих веб-страницах в качестве ссылок используют небольшие графические изображения с надписями (кнопки). Также следует упомянуть так называемые ***графические карты*** – изображения, разные части которых являются ссылками на разные ресурсы. В любом случае указатель ссылки должен подсказывать пользователю, какой именно ресурс откроется при выборе данной ссылки.

Ссылка может указывать на другую страницу того же сайта. Тогда при задании адреса ссылки следует указать относительный путь к документу назначения. Если ссылка указывает на внешний сайт, то в качестве адреса ссылки указывают полный URL.

Ссылка может указывать не только на веб-страницу, но и на другой ресурс, например, на адрес электронной почты. При выборе данной ссылки на компьютере пользователя автоматически загрузится такая программа, как почтовый клиент, которая создаст бланк нового письма с указанным адресом. Таким образом, данная ссылка представляет собой простейшее средство организации обратной связи с посетителями сайта. Адресная часть такой ссылки выглядит следующим образом: **mailto:адрес_электронной_почты.**

Наконец, ссылки могут указывать и на документы другого типа (не являющиеся веб-страницами), например, на документ Microsoft Word или на архив WinZip. В таком случае указывают относительный путь к документу (если этот документ находится внутри папки сайта) или полный URL (если этот документ хранится на внешнем сервере). При выборе данной ссылки на компьютере пользователя появится окно с предложением открыть данный файл или сохранить его на диске. Следует помнить, что для просмотра файла на компьютере пользователя должна быть установлена соответствующая программа.

Отметим, что во всех рассмотренных случаях для создания ссылки нужно лишь знать адрес ресурса назначения и совершенно не требуется редактировать документ назначения. Исключение составляют лишь так называемые ссылки на закладки – ссылки, которые открывают документ назначения в определенном месте (например, в конце). В таком случае прежде всего в документе назначения в нужном месте создается закладка, которой присваивается имя (имя закладки не должно содержать русских букв). Затем в исходном документе создается ссылка, в адресной части которой указывается относительный или абсолютный (в зависимости от ситуации) адрес документа назначения и имя закладки.

Таблицы. Использование таблиц для компоновки веб-страниц. ***Таблицы*** на веб-страницах могут использоваться как ***метод представления данных***. Таблицы можно форматировать: использовать выравнивание данных, менять размер, задавать параметры границ ячеек, применять заливку, объединять или разбивать ячейки и т. д.

Однако у таблиц на веб-страницах есть еще одна область применения. Они могут использоваться как ***средство компоновки страниц***. Это связано с тем, что вследствие некоторых особенностей языка HTML нередко достаточно сложно расположить объекты на странице друг относительно друга требуемым образом. В таком случае с помощью таблицы создается «каркас» страницы – ячейки таблицы указы-

вают области, в которых должны быть размещены соответствующие объекты. Затем простыми приемами форматирования снимают отображение границ ячеек – и при просмотре в браузере страница выглядит нужным образом.

Работая с таблицами (особенно если таблицы используются для компоновки страниц), следует помнить, что размер таблиц (как и графических изображений) можно указывать в пикселах или процентах от ширины родительского элемента. Для того чтобы компоновка страницы не зависела от разрешения монитора пользователя, размер таблицы (и ее ячеек) необходимо указывать в процентах от ширины экрана.

Формы. Нередко на страницах можно встретить и такие динамические объекты, как формы, которые посетитель может заполнить и отправить данные владельцам сайта. В форме могут использоваться следующие элементы: текстовое поле (для ввода произвольного текста), поле ввода пароля, переключатели и списки с единичным выбором (для выбора одного из предложенных вариантов), флажки и списки с множественным выбором (для выбора нескольких из предложенных вариантов), кнопка для очистки полей ввода, кнопка для отправки данных.

Порядок работы с формой следующий. Посетитель заполняет форму и отправляет данные на сервер. Программа на сервере обрабатывает полученные данные. Обработка данных может заключаться в записи данных в указанный файл, сохранении в базе данных, передаче другой программе. Затем программа генерирует и отсылает посетителю новую страницу.

1.5. Средства создания веб-страниц. Возможности редакторов визуального проектирования. Просмотр веб-страниц в браузерах

Средства создания веб-страниц. Веб-страница – это документ на языке HTML. Для записи HTML-кода вручную можно использовать простейшие текстовые редакторы, например Блокнот. Код HTML записывается в Блокноте вручную, а при сохранении его необходимо указать расширение документа: **.htm** или **.html**.

Такой способ создания веб-страниц подразумевает глубокое знание языка HTML. Кроме того, он позволяет лишь создавать отдельные страницы. Далее разработчик должен вручную выполнить связывание страниц гиперссылками, проектирование физической и логической структуры сайта, копирование всех необходимых файлов в папку сайта, проверку гиперссылок и правильности записи путей к мульти-

медийным объектам, а также многие другие операции, связанные с работой с сайтом в целом.

Существуют специализированные программы для записи HTML-кода – ***HTML-редакторы***. Большинство современных HTML-редакторов позволяют создавать веб-страницы в визуальном режиме: проектируя страницу, мы сразу же видим ее практически так, как ее будут видеть в браузере будущие посетители. Соответствующий HTML-код генерируется такими программами автоматически, без участия разработчика. Таким образом, нет необходимости записывать HTML-код вручную (тем не менее без знания основ HTML нередко не удается обойтись и в данном случае). Примером такого редактора визуального проектирования веб-страниц является программа Adobe Dreamweaver.

Для создания сайтов также можно использовать **CMS** (Content Management System – система управления содержимым), представляющую собой программное обеспечение, которое устанавливается на сервере и позволяет разработчику управлять сайтом. CMS могут включать готовые решения для создания новостей, форумов, досок объявлений, фотогалерей и др., а также шаблоны для сайтов. Примеры таких систем: Joomla, WordPress, Drupal.

Возможности редакторов визуального проектирования. Отличительной особенностью большинства редакторов визуального проектирования является то, что эти программные средства позволяют не только проектировать отдельные страницы, но и работать с сайтом в целом. Так, средствами редакторов визуального проектирования можно определить корневую папку сайта, создать внутри нее нужные подкаталоги, указать домашнюю страницу сайта. Многие такие редакторы позволяют задать логическую структуру сайта, т. е. построить его карту.

Создавая отдельные страницы, можно работать со всеми элементами, о которых шла речь выше: указывать заголовки страниц, задавать цветовую разметку страницы, фоновое изображение и фоновый звук, вводить и форматировать текст, создавать гиперссылки, вставлять и форматировать таблицы, списки, горизонтальные линии. При вставке мультимедийных объектов не нужно предварительно копировать соответствующий файл в корневую папку сайта: редактор визуального проектирования сам напомнит об этом. Кроме того, в HTML-коде будет указан именно относительный путь к объекту. То же касается и гиперссылок на страницы сайта и документы другого типа.

Если некоторые документы сайта необходимо переименовать или переместить, для этого также следует использовать редактор визуального проектирования. В таком случае ссылки на эти документы будут автоматически изменены. Следует иметь в виду, что если документ сайта переименовать или переместить средствами других программ (например, приложения Проводник), то автоматического обновления ссылок не произойдет.

Наконец, средствами редакторов визуального проектирования можно опубликовать уже готовый сайт (конечно, если вам известен адрес сервера и пароль для доступа к его подкаталогам).

Просмотр страниц в браузерах. Создавая веб-страницы, нужно помнить о том, что пользователи Интернета будут просматривать их с помощью программ-браузеров. Уже упоминалось о том, что существуют неграфические браузеры, например текстовые или речевые, которые не отображают мультимедийные объекты. Но даже привычные нам графические браузеры могут неодинаково интерпретировать HTML-код, например в том случае, если разработчик сайта использует приемы форматирования, не рекомендуемые действующим стандартом HTML. Поэтому создавая страницы, разработчик должен периодически просматривать их во всех наиболее популярных браузерах.

1.6. Методы реализации динамических веб-страниц

До сих пор речь шла о так называемых статических веб-страницах – страницах, содержимое которых остается неизменным. Однако в WWW часто можно встретить и динамические страницы – страницы, содержимое которых может меняться в зависимости от действий пользователя, информации из баз данных или внешних условий. Для создания динамических страниц используются возможности языков программирования. Программный код может исполняться на сервере (серверные технологии) или загружаться вместе со страницей и исполняться на компьютере пользователя (клиентские технологии).

Язык JavaScript можно использовать для написания так называемых скриптов. Это программы, которые могут записываться внутри документа HTML или храниться как отдельный файл и подключаться к документу HTML. Обрабатывает скрипты браузер пользователя. Скрипты широко используются для придания интерактивности веб-страницам, они могут анимировать элементы на странице, отображать текущие дату и время, позволяют работать с формами и многое другое.

Существуют библиотеки готовых скриптов, таким образом, работать с ними может практически любой пользователь.

Технология Flash предназначена для создания компьютерной анимации. Ее преимущество состоит в том, что создание Flash-объектов не требует знания языков программирования, а происходит в визуальном режиме с использованием программы Adobe Flash. Для того чтобы компьютер пользователя мог обрабатывать подобные объекты, инсталлируется надстройка к браузеру Adobe Flash Player. Инсталляция такой надстройки может произойти автоматически, когда пользователь впервые загружает Flash-объект с любой веб-страницы.

Язык PHP (первоначально расшифровывался как Personal Home Pages – личные домашние страницы; в настоящее время – Hypertext Preprocessor – гипертекстовый препроцессор). На этом языке создают PHP-сценарии, обрабатываемые на сервере. Одним из направлений применения PHP является создание динамических веб-страниц на основе информации из баз данных. Используя данный язык, можно организовать счетчик посещений, статистику обращений к разделам сайта, работать с формами, создавать чаты, форумы, гостевые книги и др.

Следует заметить, что редакторы визуального проектирования позволяют создавать некоторые динамические объекты в визуальном режиме, без прямого использования языков программирования. Счетчик посещений, дата последнего изменения страницы, временный рисунок (рисунок, который присутствует на странице лишь некоторое время, а затем замещается другим), анимационный эффект при переходе к другой странице, динамические эффекты для текста и гиперссылок (например, изменение способа форматирования текста, если пользователь выполняет щелчок мышью или указывает на него мышью), полиморфная кнопка (кнопка, которая меняет вид, когда пользователь проводит по ней мышью), формы – эти и другие объекты можно создавать средствами редакторов визуального проектирования. Соответствующие активные компоненты генерируются автоматически и либо вставляются в документ HTML, либо хранятся в отдельных файлах, прикрепленных к веб-странице.

1.7. Этапы создания веб-сайта

Выделим основные этапы создания сайта:

- выбор темы и информационного наполнения;
- проектирование логической структуры;
- проектирование физической структуры;

- создание отдельных страниц и установка связей между ними;
- тестирование;
- публикация.

Разрабатывая сайт, следует в первую очередь ориентироваться на удобство пользователя. Необходимо учитывать возможные ограничения аппаратных и программных средств пользователя, стремиться к минимальному объему страниц (для скорейшей загрузки), а также использовать корректный дизайнерский стиль. Выполнение этих простых правил позволит создать по-настоящему полезный и посещаемый сайт.

Контрольные вопросы

1. Что такое карта сайта?
2. Что представляет собой корневая папка сайта? Можно ли для хранения документов сайта создавать внутри корневой папки подкаталоги?
3. Как должен называться HTML-документ, задающий домашнюю страницу сайта, и где он должен храниться?
4. Какие элементы может содержать веб-страница?
5. Что представляет собой заголовок веб-страницы?
6. Чем отличаются относительный и абсолютный пути к документу? Почему при указании источника графического изображения используют относительный путь?
7. Для чего используются таблицы на веб-страницах?
8. Назовите основные средства создания веб-страниц. Приведите примеры.
9. Перечислите основные технологии реализации динамических веб-страниц.
10. Перечислите основные этапы создания веб-сайтов.

Лекция 2. СОЗДАНИЕ ВЕБ-СТРАНИЦ СРЕДСТВАМИ ЯЗЫКА HTML

Цель: получить представление о назначении и возможностях языка HTML, изучить синтаксис языка, ознакомиться с правилами записи основных тегов и их атрибутов.

2.1. Веб-страница как документ HTML.
2.2. Синтаксис языка HTML.

2.3. Версии языка HTML.
2.4. Структура документа HTML.
2.5. Цветовая разметка документа. Цвет в HTML.
2.6. Разметка текста.
2.7. Форматирование текста.
2.8. Графика в HTML.
2.9. Гиперссылки.
2.10. Таблицы.
2.11. Комментарии.
2.12. Служебная область HTML-документа. Заголовок веб-страницы. метатеги.

2.1. Веб-страница как документ HTML

Язык **HTML** (HyperText Markup Language – язык гипертекстовой разметки) является языком написания веб-страниц. Веб-страница представляет собой документ HTML. Этот документ может содержать:

- текст и описание его форматирования;
- определение структуры и формата страницы: может задаваться, например, деление текста на строки и абзацы, заголовок страницы, поля, цветовая разметка страницы, фоновое изображение и фоновый звук;
- указание путей к мультимедийным объектам (напомним, что сами мультимедийные объекты хранятся в отдельных файлах), а также способ их форматирования;
- таблицы;
- гиперссылки;
- активные компоненты или ссылки на документы, содержащие активные компоненты.

Интерпретирует код HTML программа-браузер, поэтому при создании веб-страниц на языке HTML необходимо как можно чаще тестировать созданный документ, открывая его в браузере.

2.2. Синтаксис языка HTML

Документ HTML содержит объекты всего двух типов: текст и так называемые теги (управляющие конструкции). Именно с помощью тегов задается форматирование текста, структура и формат страницы, вставляются и форматируются мультимедийные объекты, таблицы, гиперссылки и прочие элементы веб-страниц.

У каждого тега есть свое уникальное имя. Для того чтобы теги отличались от обычного текста, имена их заключают в угловые скобки: **<имя_тега>**. Имя тега может содержать латинские буквы и цифры. При записи имен тегов регистр не имеет значения.

Все теги языка HTML можно разделить на **парные** и **непарные**. Парные теги окружают тот фрагмент страницы, на который распространяется их действие.

Поясним это на примере. Пусть в предложении «Браузер – это программа для просмотра веб-страниц.» слово «Браузер» необходимо выделить полужирным начертанием. Тег, задающий полужирное написание, носит имя **<b>**. Таким образом, соответствующий код HTML должен выглядеть так:

```
<b>Браузер</b> – это программа для просмотра веб-страниц.
```

Обратите внимание на то, что слово «Браузер» окружено парой тегов **<b>**. Первый тег (открывающий) указывает начало фрагмента, на который должно распространяться действие тега, второй тег (закрывающий) – конец этого фрагмента. В закрывающем теге сразу после угловой скобки (<) необходимо ставить знак «/». Браузер интерпретирует этот код HTML так:

Браузер – это программа для просмотра веб-страниц.

Существуют и непарные теги, присутствие которых в определенном месте документа указывает на то, что здесь должен появиться некий объект. Для непарных тегов закрывающий тег отсутствует. Так, тег **<img>** указывает на то, что в данной области документа должно появиться графическое изображение.

Большинство тегов имеет свой набор **атрибутов** (параметров). Атрибуты уточняют действие тегов. Как правило, использование атрибутов необязательно. Для парных тегов атрибуты указывают в открывающем теге. Атрибуту может присваиваться заданное или произвольное значение. Для одного и того же тега допускается использование нескольких атрибутов, разделенных пробелами.

Так, тег **<img>** имеет ряд атрибутов: **src** указывает путь к графическому файлу – источнику изображения, **width** и **height** – ширину и высоту изображения (в пикселах или процентах), **alt** – альтернативный текст и др. Например:

```
<img src="../images/ris.jpg" width=200 height=150>
```

При интерпретации кода HTML браузер отобразит изображение из указанного графического файла шириной 200 и высотой 150 пикселов. Атрибут **alt** не задан, значит, альтернативный текст отсутствует.

Регистр имен тегов и атрибутов может быть любым. Так, тег вставки графического изображения можно записать как **<img>**, **<IMG>**, **<Img>** и т. д.

Наличие и количество пробелов между тегом и текстом или соседними тегами не имеет значения. Значения атрибутов необязательно заключать в кавычки, если они содержат только буквы, цифры, точки и дефисы.

На один и тот же фрагмент страницы может распространяться действие сразу нескольких тегов. Так, пусть фрагмент текста необходимо выделить полужирным начертанием и курсивом. Тег курсива **<i>**. Тогда код HTML записывают следующим образом:

```
<b><i>фрагмент текста</i></b>
```

Необходимо соблюдать порядок вложенности тегов. Соответствующие конструкции должны вкладываться друг в друга, не пересекаясь. Запись, подобная приведенной ниже, является неверной.

```
<b><i>фрагмент текста</b><i>
```

Поэтому многие браузеры могут неправильно интерпретировать такой HTML-код.

Следует учитывать и тот факт, что браузеры не проверяют написания кода HTML. Неправильно записанный и нераспознанный браузером фрагмент кода игнорируется. При этом никакого сообщения об ошибке браузер выдавать не будет. Таким образом, еще раз напомним о необходимости периодического просмотра документа HTML в браузере.

Примечание. Основные теги и их атрибуты для оформления HTML-документов приведены в прил. 1.

2.3. Версии языка HTML

Версии языка HTML разрабатываются Консорциумом Всемирной паутины (World Wide Web Consortium, W3C) – организацией, разрабатывающей для Интернета единые принципы и стандарты (рекомендации), которые внедряются производителями программных продуктов и оборудования.

На момент написания курса лекций действующим стандартом является HTML 4.01 (принят в 1999 году). В более ранних версиях HTML допускалось использование тегов, определяющих визуальное

представление документа (цвет, выравнивание, гарнитура шрифта и др.). В HTML 4.01 представление отделено от структуры, многие теги объявлены нерекомендуемыми к использованию. Это означает, что браузер может и не поддерживать данный тег.

В 2000 году был принят стандарт XHTML 1.0 (Extensible HyperText Markup Language – расширяемый язык разметки гипертекста), в котором действуют синтаксические правила языка XML (Extensible Markup Language – расширяемый язык разметки). В XML можно создавать собственные теги и формировать структуру документа. XHTML имеет более строгие правила верстки. Так, теги следует записывать только строчными буквами, значения атрибутов обязательно заключать в кавычки, все теги, в том числе непарные, обязательно должны быть закрыты (например, <img />). Предполагалось, что XHTML постепенно вытеснит HTML.

Далее W3C приступил к разработке XHTML 2.0 – принципиально нового языка, не совместимого с предыдущими версиями HTML. В 2009 году было объявлено, что работа над XHTML 2.0 не будет продолжена.

В настоящее время ведутся работы над HTML5. В HTML5 введен ряд тегов для описания структуры страниц (например, заголовок, подвал, панель навигации и пр.). Значительно упрощена вставка аудио и видео. Есть возможность создавать графические элементы. Появились новые широкие возможности для создания веб-приложений.

HTML5 обратно совместим, т. е. страницы, созданные в соответствии с более ранними стандартами, также будут корректно отображаться. Хотя на момент написания настоящего курса лекций HTML5 не был утвержден в качестве стандарта, многие производители браузеров уже начали внедрять его поддержку.

Чтобы браузер понимал, согласно какому стандарту HTML или XHTML отображать веб-страницу, в первой строке HTML-документа записывают элемент **<!DOCTYPE>**, указывающий DTD (Document Type Definition – описание типа документа). Ниже приведены примеры записи элемента <!DOCTYPE>.

Строгий синтаксис HTML:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

Переходный синтаксис HTML:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

Строгий синтаксис XHTML:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

Для всех документов в HTML5:

<!DOCTYPE html>

2.4. Структура документа HTML

Общий вид документа HTML следующий:

```
<html>
<head>
служебная область документа
</head>
<body>
содержательная область документа
</body>
</html>
```

Итак, документ HTML начинается и заканчивается тегами **<html>**. Служебная область документа ограничивается парным тегом **<head>**. Она содержит заголовок страницы (это единственные данные из этой области, которые увидит пользователь при просмотре страницы в браузере), а также информацию для браузеров и поисковых систем, которые будут индексировать страницу.

Собственно содержимое страницы располагается внутри парного тега **<body>**. Именно о содержательной части HTML-документа и пойдет речь далее.

2.5. Цветовая разметка документа. Цвет в HTML

Для задания цветовой схемы документа используют следующие атрибуты тега **<body>**:

- **bgcolor** – цвет фона документа;
- **background** – фоновое изображение;

- **text** – цвет текста;
- **link** – цвет непросмотренной ссылки;
- **vlink** – цвет просмотренной ссылки;
- **alink** – цвет просматриваемой ссылки.

Значение атрибута **background** – адрес документа, в котором содержится фоновое изображение. Значения всех остальных перечисленных атрибутов – цвета соответствующих элементов.

Цвет в HTML можно задать двумя способами. Первый способ – указать название цвета. В HTML есть названия для 256 цветов, но не все браузеры распознают эти названия. Однако названия основных 16 цветов способны распознать все графические браузеры, и лучше использовать только эти цвета. Приведем их названия:

black – черный, **silver** – светло-серый, **gray** – серый, **white** – белый, **maroon** – бордовый, **red** – красный, **purple** – фиолетовый, **fuchsia** – лиловый, **green** – зеленый, **lime** – желто-зеленый, **olive** – оливковый, **yellow** – желтый, **navy** – темно-синий, **blue** – синий, **teal** – сине-зеленый, **aqua** – голубой.

Цвет	Символьное обозначение	Цифровое обозначение
Черный	BLACK	#000000
Синий	BLUE	#0000FF
Темно-синий	NAVY	#000080
Зеленый	GREEN	#008000
Сине-зеленый	TEAL	#008080
Желто-зеленый	LIME	#00FF00
Голубой	AQUA	#00FFFF
Бордовый	MAROON	#800000
Фиолетовый	PURPLE	#800080
Оливковый	OLIVE	#808000
Серый	GRAY	#808080
Светло-серый	SILVER	#C0C0C0
Красный	RED	#FF0000
Лиловый	FUCHSIA	#FF00FF
Желтый	YELLOW	#FFFF00
Белый	WHITE	#FFFFFF

Если ни один из перечисленных цветов не подходит, необходимо определить цвет, используя схему RGB.

Схема RGB (Red-Green-Blue – красный-зеленый-синий) предполагает, что цвет создается путем смешения в определенных пропорциях красного, зеленого и синего цветов. Интенсивность каждого компонента записывается как двузначное шестнадцатеричное число. Например, цвет A26720 означает, что взято достаточно много красного цвета, меньше зеленого, еще меньше синего. В результате получится терракотовый цвет.

Итак, фрагмент кода HTML

```
<body bgcolor=maroon text="#256AC1">
```

означает, что фон документа будет бордовым, а цвет текста – синим с примесью зеленого и небольшой примесью красного. Обратите внимание на то, что если цвет задается по схеме RGB, то перед значением цвета необходимо ставить символ **#** и обязательно заключать значение атрибута в кавычки.

Атрибуты цветовой схемы не рекомендуются к использованию в спецификации HTML 4.01, хотя и поддерживаются в настоящее время большинством браузеров. Для задания цветовой разметки рекомендуется использовать CSS (Cascading Style Sheets – каскадные таблицы стилей).

2.6. Разметка текста

Рассмотрим теги, используемые для структурной разметки текста.

Даже такие простейшие приемы, как деление текста на строки и абзацы, на языке HTML описываются с помощью специальных тегов. При интерпретации документа HTML браузер никак не реагирует на то, каким образом в коде HTML текст разбит на строки. Деление на строки происходит в соответствии с шириной окна. Если необходимо, чтобы в определенном месте текста произошел перенос строк, следует использовать тег **
**.

Код HTML	Отображение в браузере
Текст можно разбить на строки произвольно	Текст можно разбить на строки произвольно

В примере перенос строк происходит лишь там, где присутствует тег **
**. Заметим, что при интерпретации документа браузером несколько пробелов между словами также игнорируются и превращаются в один.

Для создания нового абзаца используют парный тег **<p>**. Абзацы могут отделяться один от другого увеличенным интервалом.

Фрагменты текста можно объявить заголовками структурных элементов текста. В HTML можно создавать заголовки шести уровней. Для задания заголовков 1–6-го уровней служат парные теги **<h1>**, **<h2>**, ..., **<h6>**. Текст, объявленный заголовком того или иного уровня, при просмотре в браузере будет отформатирован определенным образом, а также этому фрагменту текста будет придан больший вес при индексации поисковыми системами.

2.7. Форматирование текста

Теги физического форматирования текста определяют формат отображения текста в графических браузерах. Все эти теги являются парными. Так, тег **<b>** задает полужирное начертание, **<i>** – курсив, **<u>** – подчеркивание, **<sub>** – нижний индекс, **<sup>** – верхний индекс.

Для определения параметров шрифта используется тег **<font>**. Атрибуты тега **<font>**:

- **size** – устанавливает ***размер шрифта***, который будет использоваться текстом, содержащимся в пределах тега **<font>**. Можно задать абсолютный размер шрифта, указав целое число от 1 до 7. Для шрифта можно также указывать ***относительный размер***, присваивая атрибуту целое число со знаком (например, это может быть size="+1" или size="–2");
- **color** – указывает ***цвет***, которым будет выделен данный фрагмент текста. Цвета задаются в виде RGB-значения с шестнадцатеричной нотацией, либо выбирается символьное значение одного из стандартных цветов;
- **face** – задает ***гарнитуру шрифта***, например, face=arial.

Теги логического форматирования текста обозначают структурные типы своих текстовых фрагментов, делают акцент на логическом, а не на визуальном выделении текста.

Например, полужирное выделение обычно применяют для новых терминов, для выделения особо важных фрагментов текста и т. д. Однако использование тега **<b>** приведет лишь к тому, что фрагмент тек-

ста будет выделен в графическом браузере. Текстовой, речевой браузер, а также поисковые системы «не догадаются» о том, что данный фрагмент текста более важен, чем остальные. В то время как использование тегов логического форматирования позволит графическому браузеру выделить фрагмент форматированием, текстовой браузер может выделить фрагмент, увеличив интервал между строками или символами, речевой браузер изменит темп, интонацию прочтения фрагмента и т. д. Наконец, поисковые машины при индексации страницы придадут фрагменту текста, помеченному тегами логического форматирования, больший вес.

На данный момент существуют следующие теги логического форматирования текста:

- **<cite>** – цитата (отметка цитат, названий источников). В графических браузерах обычно отображается курсивом;
- **<del>** – удаленный текст. В графических браузерах обычно отображается перечеркнутым текстом. Атрибут **datetime** задает дату удаления;
- **<dfn>** – определение. В графических браузерах обычно отображается курсивом;
- **** – выделение важных фрагментов текста. В графических браузерах обычно отображается курсивом;
- **** – выделение важных фрагментов текста. В графических браузерах обычно отображается полужирным шрифтом;
- **<abbr>** – аббревиатура. Атрибут **title** задает расшифровку аббревиатуры. В браузере расшифровка обычно выводится в качестве всплывающей подсказки при наведении указателя мыши на аббревиатуру.

2.8. Графика в HTML

За вставку графического изображения в документ отвечает непарный тег **<img>**. Обязательный атрибут **src** указывает адрес графического файла. Например:

```
<img src="../images/ris.gif'>
```

Некоторые другие атрибуты тега <img>:

- **width** и **height** – указывают ширину и высоту в пикселах или процентах от размеров родительского элемента;
- **alt** – значение этого атрибута есть альтернативный текст.

2.9. Гиперссылки

Для создания гиперссылки необходимо окружить указатель гиперссылки – текст или изображение – открывающим и закрывающим тегами **<a>**. Адресная часть ссылки задается атрибутом **href** тега **<a>**. Этому атрибуту необходимо присвоить значение, соответствующее адресу ресурса назначения.

Например:

```
Зайдите на сайт <a href="http://www.baa.by">БГСХА</a>
```

Здесь фрагмент текста «БГСХА» размещается между парой тегов <a>, значит, данный текст является гиперссылкой. Значение атрибута **href** – URL домашней страницы сайта БГСХА. Следовательно, при выборе данной ссылки будет загружаться именно этот ресурс.

Для создания ссылки на закладку в документе назначения в нужном месте записывают тег **<a>** с атрибутом **name**, который задает имя закладки. Имя закладки может содержать латинские буквы и цифры. Например, пусть документом назначения является документ **news.htm**, причем при переходе по ссылке пользователь должен видеть конец документа. Тогда в конце документа **news.htm** записывают следующий код:

```
<a name=end> </a>
```

В данном случае имя закладки **end**. Далее при создании ссылки в исходном документе конструкция **<a>** будет выглядеть следующим образом:

```
<a href="news.htm#end">... </a>
```

Обратите внимание на то, что адрес документа назначения и имя закладки разделены символом # (без пробелов). Заметим, что если ссылка указывает на закладку в текущем документе, указывать путь к документу необязательно.

2.10. Таблицы

Теперь рассмотрим процесс создания и форматирования таблиц в HTML. Начинается и заканчивается таблица открывающим и закрывающим тегами **<table>**. Далее записывается столько парных тегов **<tr>**, сколько строк должно быть в таблице. Между каждой парой тегов **<tr>** записывается столько парных тегов **<td>**, сколько ячеек должно быть в данной строке, причем каждая строка должна состоять

из одинакового количества ячеек. Вместо тегов ячеек **<td>** можно использовать теги **<th>**, в таком случае эти ячейки будут объявлены заголовками. Данные в этих ячейках будут автоматически выделены полужирным начертанием и выровнены по центру.

Ниже представлена таблица на языке HTML (назначение атрибутов тега **<table>** мы обсудим позднее).

```
<table border cellspasing=0>
<tr>
<th>День недели</th>
<th>Тема занятий</th>
</tr>
<tr>
<td>Понедельник</td>
<td>Навигация в WWW</td>
</tr>
<tr>
<td> Вторник</td>
<td>Поиск информации в Интернете<td> </tr>
</table>
```

Примерно так эта таблица будет выглядеть в браузере:

День недели	**Тема занятий**
Понедельник	Навигация в WWW
Вторник	Поиск информации в Интернете

Сразу после тега **<table>** можно записать парный тег **<caption>**, который определяет заголовок таблицы.

Рассмотрим некоторые приемы форматирования таблицы с использованием атрибутов тега **<table>**:

- **border** – задает ширину границы в пикселах. Следует помнить, что отсутствие атрибута **border** означает отсутствие отображения границ таблицы. Этот вариант используют, если таблица нужна для компоновки станицы. Если не присвоить атрибуту border никакого значения (см. пример выше), ширина границы будет равна 1 пикселу;
- **cellspasing** – расстояние между границами ячеек в пикселах. Если атрибут отсутствует, то между границами ячеек появится зазор величиной в 1 пиксел;

• **cellpadding** – расстояние между границей ячейки и данными в ячейке;

• **width** – ширина таблицы (в пикселах или процентах от размеров родительского элемента).

2.11. Комментарии

Для добавления в HTML-код комментариев используется следующая запись:

```
<!-- текст комментария -->
```

Текст комментария не отображается при просмотре страницы в браузере. С помощью комментариев можно делать заметки в документе, давать дополнительные разъяснения в случае, если страница создается коллективом разработчиков, и т. д. Например, комментарий-напоминание:

```
<p>Молекула воды состоит из одного атома кислорода и двух атомов водорода.</p>
<!-- Найти и добавить сюда изображение молекулы воды -->
```

2.12. Служебная область HTML-документа. Заголовок веб-страницы. Метатеги

Заголовок веб-страницы является единственным элементом в служебной области HTML-документа, который будет отображаться при просмотре страницы в браузере. Заголовок записывается внутри парного тега **<title>**.

В служебной области HTML-документа также могут находиться так называемые **метатеги**. Каждый тег **<meta>** имеет только два, но обязательных атрибута: атрибут **name** или **http-equiv** плюс атрибут **content**. Значения атрибута content зависят от значений атрибутов **name** и **http-equiv**. Теги **<meta>** с атрибутом **name** в основном содержат информацию для поисковых машин. Теги **<meta>** с атрибутом **http-equiv** в основном содержат информацию для браузеров.

Рассмотрим некоторые возможные значения атрибута **name** тега **<meta>** и соответствующие им значения атрибута **content**:

• **name=description**, тогда атрибут **content** указывает краткое описание сайта. Например:

```
<meta name="description" content="Страничка для любителей тенниса">.
```

• **name**=**keywords**, тогда атрибут **content** указывает ключевые слова (через запятую). Например:

```
<meta name="keywords" content="спорт, теннис, sport, tennis">
```

• **name**=**site-created**, тогда атрибут **content** указывает дату создания ресурса в формате МЕСЯЦ-ДЕНЬ-ГОД. Например:

```
meta name="site-created" content="l1-04-2012">
```

Это означает, что ресурс был создан 4 ноября (а не 11 апреля) 2012 года.

• **name**=**expires**, тогда атрибут **content** указывает предполагаемую дату закрытия ресурса в формате МЕСЯЦ-ДЕНЬ-ГОД. Например:

```
<meta name ="expires" content="01-01-2016">
```

• **name**=**revisit**, тогда атрибут **content** указывает количество дней (от 1 до 30), по прошествии которых необходимо переиндексировать ресурс (для часто обновляемых страниц). Например:

```
<meta name="revisit" content="7 days">
```

Отметим, что слово **days** в значении атрибута **content** обязательно.

• **name**=**content-language**, тогда атрибут **content** указывает язык ресурса (на английском языке). Например:

```
<meta name="content-language" content="russian">
```

• **name**=**author**, тогда атрибут **content** указывает имя автора ресурса. Например:

```
<meta name="author" content="John Smith">
```

• **name**=**owner**, тогда атрибут **content** указывает имя человека (название организации), являющегося владельцем ресурса. Например:

```
<meta name="owner" content="БГСХА">
```

Теперь рассмотрим некоторые возможные значения атрибута **http-equiv** тега **<meta>** и соответствующие им значения атрибута **content:**

• **http-equiv**=**refresh**, тогда атрибут **content** указывает количество секунд, через которое необходимо перезагружать страницу или количество секунд, через которое необходимо перейти по указанному адресу. Например:

```
<meta http-equiv="refresh" content="60">
```

Это означает, что страницу нужно перезагружать каждые 60 секунд (для страниц, содержимое которых обновляется сверхчасто, например, для страниц, содержащих биржевые сводки). Или

```
<meta http-equiv="refresh" content="20; url=http://www.new.by">
```

Это означает, что через 20 секунд необходимо перейти по указанному URL. Такой вариант нужен в случае, когда давно существующий сайт меняет адрес.

• **http-equiv=content-type**, тогда атрибут **content** указывает кодировку ресурса. Например:

```
<meta http-equiv="content-type" content="text/html; charset=windows-1251">
```

Контрольные вопросы

1. Какие объекты могут содержаться внутри документа HTML?
2. Чем отличаются парные и непарные теги?
3. Каковы основные правила записи тегов и их атрибутов?
4. Какие теги определяют служебную и содержательную области документа HTML?
5. Назовите способы указания цвета в HTML.
6. Какие теги применяются для физического форматирования текста? Укажите недостатки физического форматирования текста.
7. Что представляет собой логическое форматирование текста?
8. Для чего служит тег <img>? Почему атрибут src этого тега является обязательным?
9. Как задать таблицу в HTML?
10. Для чего используются метатеги и каковы их атрибуты? Приведите примеры значений атрибутов метатегов.

Лекция 3. ОФОРМЛЕНИЕ ВЕБ-СТРАНИЦ СРЕДСТВАМИ CSS

Цель: получить представление о назначении и преимуществах использования каскадных таблиц стилей (CSS), изучить синтаксис CSS.

3.1. Каскадные таблицы стилей. Преимущества использования.
3.2. Синтаксис CSS.
3.3. Классы.
3.4. Идентификаторы.
3.5. Добавление стилей в HTML-документ: внедрение и связывание. Универсальный атрибут style.

3.6. Комментарии. Приоритеты и разрешение конфликтов.

3.7. Блочная модель в CSS.

3.1. Каскадные таблицы стилей. Преимущества использования

Для форматирования элементов веб-страницы можно применять так называемые **каскадные таблицы стилей** (Cascading Style Sheets, CSS), описывающие способы представления (отображение и расположение) элемента (или группы элементов) веб-страницы. Преимущества использования каскадных таблиц стилей состоят в следующем:

- расширяются возможности форматирования (по сравнению с HTML);
- в основной части документа отсутствуют теги физического форматирования текста, т. е. представлена лишь структурная разметка документа;
- сокращается объем HTML-документа;
- возможно единое стилевое оформление для группы веб-страниц.

Стандарты CSS, как и HTML, разрабатывает и рекомендует Консорциум Всемирной паутины (World Wide Web Consortium, W3C).

3.2. Синтаксис CSS

Технология CSS предполагает создание ***правил***, которые переопределяют отображение тегов в документе или задают пользовательский стиль. Правило состоит из ***селектора*** и ***определения*** – свойств и их значений – и записывается следующим образом:

селектор {свойство1: значение1; свойство2: значение2;...}

Селектор указывает элемент, который необходимо форматировать.

Это может быть тег (форматирование будет применено ко всем элементам данного типа), класс (форматирование можно применять выборочно к нескольким элементам) или идентификатор (форматирование можно применять только к одному элементу).

Определение указывает способ оформления. Оно заключается в фигурные скобки, внутри которых перечислены свойства и их значения. Свойства определяют параметры форматирования. Например, свойство **color** указывает цвет элемента. Свойствам присваивается заданное или произвольное (в зависимости от свойства) значение. Между свойством и значением ставят двоеточие. Регистр символов и разбиение правила на строки не имеют значения.

Например:

```
p {color: red}
```

Такое правило означает, что во всех абзацах цвет шрифта должен быть красным.

Для одного и того же селектора можно указать несколько свойств. В таком случае их нужно перечислить в фигурных скобках через точку с запятой, например:

```
p {color: red; text-weight: bold}
```

Это означает, что во всех абзацах шрифт должен быть красного цвета и полужирный. Порядок записи свойств не имеет значения.

Если для разных селекторов необходимо указать одинаковые параметры форматирования, селекторы можно сгруппировать, перечислив через запятую:

```
h1, h2, h3 {color: green; text-align: center}
```

Это означает, что заголовки 1, 2 и 3-го уровней необходимо отображать зеленым цветом и выравнивать их по центру.

Для улучшения восприятия CSS-кода принято записывать селектор и каждое свойство в отдельных строках:

селектор {
свойство1: значение1;
свойство2: значение2;
…
}

Например, приведенные ниже правила CSS для заголовков 1-го уровня являются идентичными и любой из представленных вариантов записи считается допустимым.

```
h1{color: green; font-size:20px}
```

```
h1{
color: green;
font-size: 20px
}
```

При создании правил CSS следует грамотно выбрать селектор. Если стилевые параметры планируется применить ко всем элементам определенного типа, то в качестве селектора будет использоваться **имя тега**. Если стилевые параметры следует назначить только некоторой группе элементов (элементам, принадлежащим к определенному классу), то селектором будет **.имя_класса**. Если же стилевые параметры

задаются для одного элемента с определенным именем (идентификатором), то селектором будет **#имя_идентификатора**.

3.3. Классы

Если необходимо изменить способ отображения *всех* элементов одного и того же типа, записывают правило, селектором которого является соответствующий тег. Если же форматирование нужно применить лишь к некоторым элементам, следует создать **класс**. Тогда селектором в правиле будет имя класса (оно должно содержать только латинские буквы, цифры, символы дефиса (-) и подчеркивания (_)). При записи правила для класса перед селектором ставится точка. Например:

```
.special {color: blue; font-size: 14 pt; text-align: center}
```

В данном примере описан класс **special**, задающий синий цвет шрифта, размер 14 пунктов и выравнивание текста по центру. Чтобы применить его к конкретному элементу, в соответствующий тег добавляют атрибут **class** и присваивают ему значение, соответствующее имени класса (без точки в начале):

<p class="special">

<hl class="special">

Если предполагается, что класс будет применяться только к элементам определенного типа, т. е. к конкретному тегу, то запись правила будет выглядеть так:

```
p.special {color: blue; font-size: 14 pt; text-align: center}
```

В таком случае класс **special** можно применять только к абзацам.

3.4. Идентификаторы

Идентификатор определяет уникальное имя элемента для изменения его стиля. С помощью идентификатора можно обратиться к элементу через скрипт. Имя идентификатора должно содержать только латинские буквы, цифры, символы дефиса (-) и подчеркивания (_). При записи правила для идентификатора перед селектором ставится символ #:

```
#vvedenie {font-style: italic; text-align: right}
```

Этот идентификатор можно применять только к одному элементу на странице, используя атрибут **id**, например:

```
<p id="vvedenie">
```

3.5. Добавление стилей в HTML-документ: внедрение и связывание. Универсальный атрибут style

Существуют два способа добавления стилей в HTML-документ:

1) таблица стилей задается непосредственно в HTML-документе (**внедрение**);

2) таблица стилей хранится в отдельном файле ***.css** и подключается к HTML-документу (**связывание**).

В первом случае используется тег **<style>**, который располагается в служебной области HTML-документа **<head>** и задает стилевую информацию для текущего документа. Кроме того, можно использовать атрибут **style** для определения стилевых параметров конкретного элемента документа.

Стилевая информация может храниться непосредственно в HTML-документе, в служебной области между тегами **<style>**:

```
<head>
<style>
    h2 {color: green; font-size: 20 pt}
    p.special {color: blue; font-size: 14 pt; text-align: center}
    #vvedenie {font-style: italic; text-align: right}
</style>
</head>
```

Такой способ добавления стилей называется **внедрением**. Заметим, что в этом случае стилевое форматирование работает только для одной веб-страницы. Чтобы применить такое же форматирование для других страниц, в служебных областях соответствующих HTML-документов также должен присутствовать тег **<style>** с описанием тех же правил.

Чтобы избежать многократного повторения одних и тех же правил в разных документах, таблицу стилей можно хранить в отдельном файле (с расширением **.css**). В этом случае таблица стилей присоединяется к HTML-документу с помощью тега **<link>** в служебной части документа:

```
<link href="адрес файла css" ret=" stylesheet" type="text/css">
```

Такой способ добавления стилей называется **связыванием**. В этом случае одну и ту же таблицу стилей можно присоединить к нескольким документам, что позволит быстро и однотипно форматировать веб-страницы.

Также для определения стиля конкретного элемента можно использовать атрибут **style** соответствующего тега, например:

```
<p style="color: red">...</p>
```

Здесь шрифт меняет цвет на красный только в указанном абзаце.

3.6. Комментарии. Приоритеты и разрешение конфликтов

Комментарии позволяют разъяснять то или иное правило другим разработчикам, делать заметки. Комментарии не выводятся при просмотре страницы в браузере. Их можно также добавлять, применяя конструкцию **/*... */.**

Например:

```
.def {font-size: 14 pt; text-weight: bold}
/*Это стиль для определений*/
```

Приоритеты и разрешение конфликтов. Если в документе наблюдается конфликт стилей, т. е. к одному и тому же элементу применяются противоречивые правила, браузер при интерпретации такого документа учитывает приоритеты стилей. Так, среди рассмотренных нами вариантов задания стилей приоритеты распределяются следующим образом (от высшего к низшему):

1. Атрибут style внутри тега.
2. Идентификатор.
3. Класс.
4. Тег.

Например, в служебной области документа задано правило:

```
p {color: red}
```

При этом в содержательной части для одного отдельного абзаца задан стиль:

```
<p style="color: blue">
```

В данном случае в указанном абзаце текст будет отображаться синим цветом, а в остальных – красным.

Если в результате многократного редактирования страницы и подключения различных внешних стилевых файлов для одного и того же селектора будут записаны разные правила, то применяется то, которое записано последним. Например, для тега **h1** записаны два правила:

```
h1 {color: green}
```

и ниже

```
h1 {color: black}
```

Текст заголовков 1-го уровня будет отображаться черным цветом, так как это правило указано позднее.

3.7. Блочная модель в CSS

Средствами CSS можно не только управлять такими стандартными параметрами форматирования, как цвет, фон, шрифт, начертание и пр., но также и задавать размеры и расположение элементов веб-страницы. Фактически любой элемент документа HTML (заголовки 1–6-го уровней, текстовой абзац, рисунок, таблица и т. д.) представляет собой прямоугольный блок, структура которого показана на рис. 4.

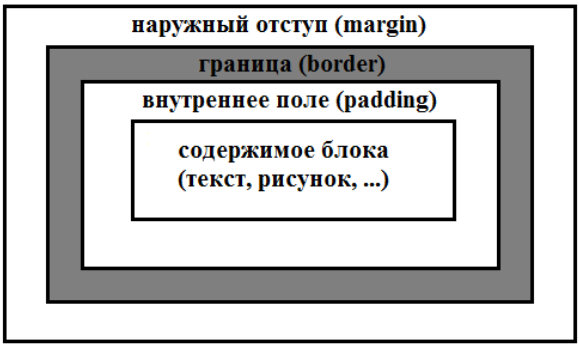

Рис. 4. Структура блока в CSS

Свойства CSS, описывающие блочную модель, представлены на рис. 5. Так, на странице блок занимает некоторую область, размеры которой определяются следующим образом:

Ширина=margin-left+border-left+padding-left+width+padding-right+ border-right+margin-right

Высота=margin-top+border-top+padding-top+height+padding-bottom+ border-bottom+margin-bottom

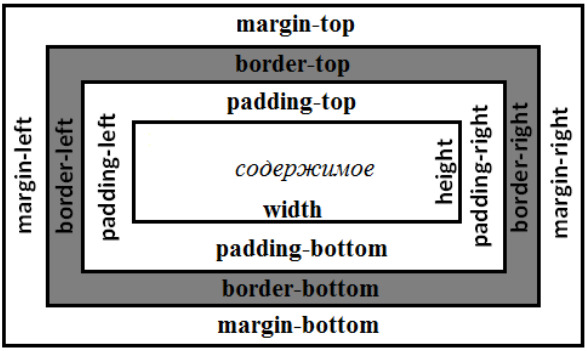

Рис. 5. Свойства CSS, описывающие блочную модель

Примечание. Основные свойства CSS приведены в прил. 2.

Контрольные вопросы

1. Каковы преимущества использования CSS для форматирования элементов веб-страницы?
2. Укажите основные принципы создания правил CSS.
3. Как записать правило, определяющее для всех абзацев на веб-странице желтый цвет шрифта?
4. Для чего используют классы?
5. Чем различаются классы и идентификаторы?
6. Назовите способы добавления стилей в документ HTML.
7. В каких случаях таблицу стилей целесообразно хранить в отдельном CSS-файле?
8. Как разрешаются конфликты при применении к одному элементу веб-страницы противоречивых правил CSS?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Брезгунова, И. В. Основы веб-проектирования: учеб.-метод. пособие (с электронным приложением) / И. В. Брезгунова, С. Н. Гринчук. – Минск: РИВШ, 2013. – 126 с.

2. Adobe Dreamweaver CS5. Официальный учебный курс / пер. М. А. Райтмана. – Москва: Эксмо, 2011. – 496 с.

3. Хеник, Б. HTML и CSS. Путь к совершенству / Б. Хеник. – Санкт-Петербург: Питер, 2011. – 336 с.

4. Роббинс, Дж. Web-дизайн: справочник / Дж. Роббинс. – Москва: Кудиц-образ, 2008. – 816 с.

5. Учебник HTML и CSS. Создай свой веб-сайт [Электронный ресурс]. – Режим доступа: http://ru.html.net. – Дата доступа: 28.12.15.

ПРИЛОЖЕНИЯ

Приложение 1

Таблица 1. **Основные теги для оформления HTML-документа**

<html>…</html> – начальный и конечный теги всего HTML-документа		
№	Атрибут	Функция
1	version=*строка*	Указание версии HTML, которая была использована для создания данного документа
<head>…</head> – начальный и конечный теги служебной области HTML-документа		
<**body**>… <**body**> – начальный и конечный теги тела документа		
№	Атрибут	Функция
1	alink=*цвет*	Установка цвета активных гипертекстовых ссылок в документе
2	background=*url*	Указание URL фонового изображения
3	bgcolor=*цвет*	Установка цвета фона документа
4	bgproperties=*значение*	Если *значение* = fixed, запрещается прокрутка фонового изображения вместе с содержимым документа
5	leftmargin=*значение*	Установка размера (в пикселах) левого поля документа
6	link=*цвет*	Установка цвета непосещенных гипертекстовых ссылок в документе
7	text=*цвет*	Установка цвета обычного текста в документе
8	topmargin=*значение*	Установка размера (в пикселах) верхнего поля документа
9	vlink= *цвет*	Установка цвета посещенных ссылок в документе
<title>…</title> – начальный и конечный теги заголовка HTML-документа		
<a>… – начальный и конечный теги, которые позволяют создать гиперссылку (атрибут **href**) или идентификатор фрагмента (атрибут **name**) HTML-документа		
№	Атрибут	Функция
1	href=*url*	Указание URL адреса целевого документа гиперссылки (необходим, если это не якорь имени)
2	methods=*список*	Задание списка методов отображения, зависящих от браузера (через запятую)
3	name=*строка*	Указание имени идентификатора фрагмента
4	rel=*связь*	Определение связи данного документа с целевым документом
5	rev=*связь*	Определение обратной связи целевого документа с данным документом
6	target=*имя*	Задание имени кадра или окна отображения обозначенного ссылкой документа
7	title=*строка*	Задание заголовка целевого документа
8	urn=*urn*	Указание не зависящего от места нахождения универсального имени ресурса для данной гиперссылки
<address>…</address> – заключенный в данные теги текст представляет собой адрес		
**** …**** – заключенный в данные теги текст будет отображаться жирным шрифтом		

Продолжение табл. 1

<table>
<tr><td colspan="3"><big>…</big> – заключенный в данные теги текст будет отображаться шрифтом большего размера</td></tr>
<tr><td colspan="3">Тег
 – разрыв текущего текстового потока и возобновление его с начала следующей строки</td></tr>
<tr><td>№</td><td>Атрибут</td><td>Функция</td></tr>
<tr><td>1</td><td>clear=поле</td><td>Задание обтекания объекта, расположение которого указано значением данного атрибута (left, right или all). При переносе на новую строку текст будет размещаться так, чтобы объект оставался видимым</td></tr>
<tr><td colspan="3"><cite> …</cite> – заключенный в данные теги текст представляет собой цитату</td></tr>
<tr><td colspan="3"><comment>…</comment> – комментарий в тексте документа. Комментарии будут видимы в любом браузере. Для всех браузеров комментарии представляются в виде <!--текст комментария--></td></tr>
<tr><td colspan="3"><dfn>…</dfn> – текст, заключенный в данные теги, форматируется как определение</td></tr>
<tr><td colspan="3"><dl>…</dl> – создание списка определений, содержащих теги <dt> и <dd></td></tr>
<tr><td colspan="3"><dd>…</dd> – задание описанной части для элемента списка определений</td></tr>
<tr><td colspan="3"><dt>…</dt> – задание описательно-условной части для элемента списка определений</td></tr>
<tr><td colspan="3">… – установка размера, цвета или гарнитуры заключенного в данные теги текста</td></tr>
<tr><td>№</td><td>Атрибут</td><td>Функция</td></tr>
<tr><td>1</td><td>color=цвет</td><td>Установка цвета заключенного в теги текста</td></tr>
<tr><td>2</td><td>face=список</td><td>Установка гарнитуры заключенного в теги текста (устанавливается первый из указанных в разделенном запятыми списке имен шрифтов)</td></tr>
<tr><td>3</td><td>size=значение</td><td>Установка размера базового шрифта. Диапазон от 1 до 7</td></tr>
<tr><td colspan="3"><Hn>…<Hn> – заключенный в данные теги текст представляет собой заголовок уровня n. Возможные значения n от 1 до 6</td></tr>
<tr><td>№</td><td>Атрибут</td><td>Функция</td></tr>
<tr><td>1</td><td>align=тип</td><td>Указание способа выравнивания заголовка: по левому краю (left, по умолчанию), по центру (center) или по правому краю (right)</td></tr>
<tr><td colspan="3">Тег <hr> – разрыв текущего текстового потока. В месте разрыва будет вставлена горизонтальная линейка</td></tr>
<tr><td>№</td><td>Атрибут</td><td>Функция</td></tr>
<tr><td>1</td><td>align=тип</td><td>Указание способа выравнивания линейки: по левому краю (left, по умолчанию), по центру (center) или по правому краю (right)</td></tr>
<tr><td>2</td><td>noshade</td><td>Запрещение использования объемного затенения при отображении линейки</td></tr>
<tr><td>3</td><td>size=n</td><td>Установка толщины линейки, равной целому числу пикселов</td></tr>
<tr><td>4</td><td>width=значение</td><td>Установка ширины линейки, равной целому числу пикселов</td></tr>
<tr><td colspan="3">… – заключенный в данные теги текст будет отображаться в курсивном начертании</td></tr>
</table>

Тег **<img>** – в текущий текстовой поток вставляется изображение		
№	Атрибут	Функция
1	alt=*текст*	Задание альтернативного текста для браузеров, не поддерживающих работу с изображениями
2	border=*n*	Установка толщины (в пикселах) обрамления изображений, содержащихся в гиперссылках
3	controls	Добавление функций управления воспроизведением встроенных видеоклипов
4	dynsrc=*url*	Задание URL адреса видеоклипа, подлежащего изображению
5	height=*n*	Задание высоты изображения в пикселах
6	hspace=*n*	Задание размещения слева и справа от изображения областей свободного пространства шириной по *n* пикселов
7	ismap	Указание того, что при использовании данного тега внутри тега **<a>** изображение выбирается с помощью мыши
8	loop=*значение*	Установка числа повторов воспроизведения видео. *Значение* может быть целым или значением *infinite*.
9	lowsrc=*url*	Указание изображения с низким разрешением, которое браузер должен загрузить первым. За ним следует изображение, заданное атрибутом **<src>**
10	src=*url*	Указание исходного URL изображения, подлежащего воспроизведению. Данный атрибут является необходимым
11	start=*начало*	Указание того, когда следует воспроизвести видеоклип (варианты: *fileopen* или *mouseover*)
12	usemap=*url*	Указание чувствительной к перемещению мыши области изображения
13	vspace=*n*	Задание размещения над изображением и под ним областей свободного пространства по *n* пикселов
14	width=*n*	Указание ширины изображения в пикселах
Тег **<link>** – в заголовке (**<head>**) документа определяется ссылка из данного документа на другой документ		
№	Атрибут	Функция
1	href=*url*	Указание URL адреса гипертекстовой ссылки целевого документа
2	methods=*список*	Задание списка методов отображения для данной ссылки, зависящих от браузера (через запятую)
3	rel=*связь*	Определение связи данного документа с целевым документом. Для Internet Explorer 3.0 rel=style означает существование внешней таблицы стилей
4	rev=*связь*	Определение обратной связи целевого документа с данным документом
5	src=*url*	Указание URL внешней таблицы стилей, которая будет использоваться для форматирования документа (*IE2* и выше)

6	title=*строка*	Задание заголовка целевого документа
7	type=*text/css*	Задание типа внешней ссылки, которая будет использоваться как внешняя каскадная таблица стилей
8	urn=*urn*	Указание для целевого документа универсального имени ресурса, не зависящего от его места нахождения
Тег **<map>** – определение чувствительной к перемещению мыши области изображения		
№	Атрибут	Функция
1	name=*строка*	Задание имени данной области. Данный атрибут является необходимым
<nobr> …</nobr> – в заключенном в данные теги тексте разрывы не допускаются		
** **…** ** – начальный и конечный теги абзаца		
№	Атрибут	Функция
1	align=*mun*	Задание способа выравнивания текста в абзаце: по левому краю (*left*), центру (*center*) или по правому краю (*right*)
<pre>…</pre> – заключенный в данные теги текст будет отображаться так, как он был отформатирован предварительно, без обработки, с точным соблюдением переносов строк и интервалов		
№	Атрибут	Функция
1	width=*n*	Браузер будет размещать текст так, чтобы в строке умещалось (если возможно) *n* символов
<s>…**</s>** – заключенный в данные теги текст будет отображаться перечеркнутым горизонтальной линией		
****…**** – заключенный в данные теги текст будет отображаться шрифтом меньшего размера		
<strike>…**</strike>** – заключенный в данные теги текст будет отображаться перечеркнутым горизонтальной линией		
_… – заключенный в данные теги текст будет отображаться как нижний индекс		
^{**…**} – заключенный в данные теги текст будет отображаться как верхний индекс		
<tt>…**</tt>** – заключенный в данные теги текст будет отображаться моноширинным шрифтом		

Таблица 2. **Теги для создания и форматирования таблицы**

<table>…**</table>** – начальный и конечный теги таблицы		
№	Атрибут	Функция
1	align=*позиция*	Задание способа выравнивания таблицы относительно текстового потока, в котором она находится: *left* (по левому краю) или *right* (по правому краю)
2	background=*url*	Указание фонового изображения для таблицы
3	bgcolor=*цвет*	Задание цвета фона всей таблицы
4	border=*n*	Создание обрамления с толщиной линий в *n* пикселов
5	bordercolor=*цвет*	Задание цвета обрамления всей таблицы

Продолжение табл. 2

6	bordercolordark=*цвет*	Задание темного цвета для создания эффекта тени рамки
7	bordercoloright=*цвет*	Задание светлого цвета для создания эффекта тени рамки
8	cellpadding=*n*	Задание расстояния между границами ячейки и ее содержимым в *n* пикселов
9	cellspacing=*n*	Задание интервала между ячейками таблицы в *n* пикселов
10	frame=[*void*|*above*| *below*|*hsides*|*lhs*|*rhs*| *vsides*|*box*|*border*]	Указание способа отображения рамки таблицы: *void* – подавление отображения всей рамки таблицы; *box* и *border* – отображение всей рамки; *hsides* – отображение горизонтальных составляющих рамки, *vsides* – вертикальных; *lhs* – отображение левой боковой стороны рамки, *rhs* – правой
11	hspases=*n*	Задание размещения слева и справа от таблицы областей свободного пространства заданной ширины в пикселах
12	rules=[*all*|*cols*| *groups*|*none*| *rows*]	Выключение (*none*) или включение режима отображения разграничительных линий между ячейками таблицы: по столбцам (*cols*), строкам (*rows*), группам (*groups*) или всем элементам (*all*)
13	vspace=*n*	Задание размещения над таблицей и под ней областей свободного пространства заданной высоты в пикселах
14	width=*n*	Установка ширины таблицы в пикселах или в процентах от ширины окна
<caption>…</caption> – теги, задающие заголовок таблицы		
<td>…**</td>** – начальный и конечный теги ячейки данных таблицы		
№	Атрибут	Функция
1	nowrap	Отключение режима автоматического распределения текста по всей ячейке. Будет отображаться лишь та часть текста, которая умещается в ячейке по всей длине
2	rowspan=*n*	Данная ячейка охватывает *n* соседних строк
3	colspan=*n*	Данная ячейка охватывает *n* соседних столбцов
4	valign=*mun*	Задание способа размещения содержимого данной ячейки по вертикали: сверху (*top*), по центру (*center*), внизу (*bottom*) или по базовой линии (*baseline*) ячейки
5	width=*n*	Задание ширины данной ячейки в пикселах или в процентах от ширины таблицы
<th>…**</th>** – начальный и конечный теги ячейки заголовка столбца таблицы		
№	Атрибут	Функция
1	align=*mun*	Задание способа выравнивания содержимого ячейки: по левому краю (*left*), по центру (*center*) или по правому краю (*right*)
2	background=*url*	Задание фонового изображения для ячейки
3	bgcolor=*цвет*	Задание цвета фона для ячейки
4	bordercolor=*цвет*	Задание цвета обрамления ячейки

Окончание табл. 2

5	bordercolordark=*цвет*	Задание темного цвета для создания эффекта тени обрамления ячейки
6	bordercololight=*цвет*	Задание светлого цвета для создания эффекта тени обрамления ячейки
7	colspan=*n*	Данная ячейка охватывает *n* соседних столбцов
8	nowrap	Отключение режима автоматического распределения текста по всей ячейке. Будет отображаться лишь та часть текста, которая умещается в ячейке по всей длине
9	rowspan=*n*	Данная ячейка охватывает *n* соседних строк
10	valign=*тип*	Задание способа размещения содержимого данной ячейки по вертикали: сверху (*top*), по центру (*center*), внизу (*bottom*) или по базовой линии (*baseline*) ячейки
11	width=*n*	Задание ширины данной ячейки в пикселах или в процентах от ширины таблицы
<tr>…</tr> – начальный и конечный теги строки ячеек в таблице		
№	Атрибут	Функция
1	align=*тип*	Задание способа выравнивания содержимого ячейки: по левому краю (*left*), по центру (*center*) или по правому краю (*right*)
2	background=*url*	Задание фонового изображения для ячейки
3	bgcolor=*цвет*	Задание цвета фона для данной строки
4	border=*n*	Создание обрамления, образованного линиями толщиной в *n* пикселов
5	bordercolor=*цвет*	Задание цвета обрамления строки
6	bordercolordark=*цвет*	Задание темного цвета для создания эффекта тени обрамления строки
7	bordercololight=*цвет*	Задание светлого цвета для создания эффекта тени обрамления строки
8	valign=*тип*	Задание способа размещения содержимого ячеек в данной строке по вертикали: сверху (*top*), по центру (*center*), внизу (*bottom*) или по базовой линии (*baseline*) ячейки

Таблица 3. **Теги для создания и форматирования списков**

** … ** – начальный и конечный теги нумерованного списка		
№	Атрибут	Функция
1	type=*значение*	Задание типа номера: A – заглавные буквы, a – строчные буквы, I – заглавные римские цифры, i – строчные римские цифры, 1 – арабские цифры (по умолчанию)
2	start=*значение*	Задание начального значения при использовании арабских цифр
3	title=*текст*	Задание всплывающей подсказки
Тег ** ** – задание элемента нумерованного или маркированного списка		
** … ** – начальный и конечный теги маркированного списка		

№	Атрибут	Функция
1	type=*значение*	Задание типа маркера: *square* – квадрат, *circle* – незакрашенный кружок, *disc* – закрашенный кружок (по умолчанию)
2	title=*текст*	Задание всплывающей подсказки

Таблица 4. **Теги для создания фреймов**

<frame>…</frame> – начальный и конечный теги для создания фрейма		
№	Атрибут	Функция
1	bordercolor=*цвет*	Установка цвета обрамления фрейма, если обрамление включено атрибутом frameborder=*yes*
2	frameborder=[*1*|*0*] или [*yes*|*no*]	Включение или выключение отображения трехмерного обрамления фрейма. По умолчанию 1 (*yes*), т. е. вставляется обрамление. Значение 0 (*no*) выключает обрамление или заменяет его обычной рамкой в зависимости от типа браузера
3	marginheight=*n*	Задание размещения над содержимым фрейма и под ним областей свободного пространства высотой по *n* пикселов
4	marginwidth=*n*	Задание размещения слева и справа от содержимого фрейма областей свободного пространства шириной по *n* пикселов
5	name=*строка*	Задание имени фрейма
6	noresize	Запрещение изменения размеров фрейма
7	scrolling=*тип*	Линейки прокрутки будут добавляться всегда (*yes*), не будут добавляться (*no*), будут добавляться, если необходимо (*auto*)
8	src=*url*	Задание URL исходного документа для данного фрейма
<noframes>…</noframes> – начальный и конечный теги, с помощью которых определяется содержимое, которое будет отображаться браузерами, не поддерживающими фреймы		
<framset>…**</framset>** – начальный и конечный теги набора фреймов		
№	Атрибут	Функция
1	border=*n*	Установка размера (в пикселах) обрамления кадров в наборе фреймов. По умолчанию толщина линий обрамления равна 5 пикселам
2	bordercolor=*цвет*	Установка цвета обрамления для набора фреймов
3	cols=*список*	Указание количества и ширины фреймов в наборе
4	frameborder=[*1*|*0*] или [*yes*|*no*]	Включение или выключение отображения трехмерного обрамления фреймов. По умолчанию 1 (*yes*) (трехмерное обрамление)
5	framespacing=*n*	Указание того, что между смежными кадрами должен быть промежуток, и задание его размера в пикселах
6	rows=*список*	Указание количества и высоты кадров в наборе

Окончание табл. 4

<iframe>…**</iframe>** – начальный и конечный теги, с помощью которых определяется плавающий фрейм, который аналогичным образом будет размещен в **<img>**		
№	Атрибут	Функция
1	align=*mun*	Задание способа выравнивания текста в ячейках столбца. Возможные значения: *center, left, right*
2	frameborder=[*1*|*0*] или [*yes*|*no*]	Включение или выключение отображения трехмерного обрамления фрейма. По умолчанию 1 (*yes*) (трехмерное обрамление)
3	height=*n*	Указание высоты фрейма в пикселах или в процентах от высоты окна
4	hspace=*n*	Задание размещения слева и справа от изображения областей свободного пространства шириной по *n* пикселов
5	marginheight=*n*	Задание размещения над содержимым фрейма и под ним областей свободного пространства высотой по *n* пикселов
6	marginwidth=*n*	Задание размещения слева и справа от содержимого фрейма областей свободного пространства шириной по *n* пикселов
7	name=*строка*	Задание имени фрейма
8	noresize	Запрещение изменения размеров фрейма
9	scrolling=*mun*	Линейки прокрутки будут добавляться всегда (*yes*), не будут добавляться (*no*), будут добавляться, если необходимо (*auto*)
10	src=*url*	Задание URL исходного документа для данного фрейма
11	vspace=*n*	Задание размещения над изображением и под ним областей свободного пространства высотой по *n* пикселов
12	width=*n*	Указание ширины фрейма в пикселах или в процентах от размера окна

Таблица 5. **Теги для написания форм**

<form>…**<form>** – начальный и конечный теги формы		
№	Атрибут	Функция
1	action=*url*	Указание URL приложения, предназначенного для обработки формы. По умолчанию используется текущий URL
2	anctype=*кодирование*	Задание способа кодирования значений элементов формы
3	method=*стиль*	Указание метода передачи параметров (*get* или *post*). По умолчанию используется *get*
4	target=*имя*	Задание целевого окна для загрузки результатов заполнения формы. Можно использовать специальные атрибуты: _bottom, _top, _parent, _self

Продолжение табл. 5

<input type=checkbox> – создание в **<form>** элемента ввода «опция»		
№	Атрибут	Функция
1	value=*строка*	Указание значения параметра, посылаемого в приложение обработки форм при выборе данного элемента формы. Этот атрибут является необходимым
2	name=*строка*	Указание имени параметра, который будет передан в приложение обработки форм при выборе данного элемента ввода. Этот атрибут является необходимым
3	checked	Отображение элемента как выбранного по умолчанию
<input type=file> – создание в **<form>** элемента ввода «выбор файла»		
№	Атрибут	Функция
1	maxlength =*n*	Указание для данного элемента максимального количества вводимых символов
2	name=*строка*	Указание имени параметра, который будет передан в приложение обработки форм при выборе данного элемента ввода. Этот атрибут является необходимым
3	size=*n*	Указание для данного элемента количества отображаемых символов
<input type=hidden> – создание в **<form>** скрытого элемента		
№	Атрибут	Функция
1	maxlength=*n*	Указание для данного элемента максимального количества вводимых символов
2	name=*строка*	Указание имени параметра, который передается в приложение обработки форм для данного элемента ввода. Этот атрибут является необходимым
3	size=*n*	Указание для данного элемента количества отображаемых символов
4	value=*строка*	Указание значения параметра, посылаемого в приложение обработки форм
<input type=image> – создание в **<form>** элемента ввода «изображение»		
№	Атрибут	Функция
1	align=*mun*	Задание размещения изображения по вертикали относительно текста данного элемента формы: вверху (*top*), посередине (*middle*) или внизу (*bottom*)
2	name=*строка*	Указание имени параметра, который будет передан в приложение обработки форм при выборе данного элемента ввода. Этот атрибут является необходимым
3	src=*url*	Указание исходного URL адреса изображения. Данный атрибут является необходимым
<input type=password> – создание в **<form>** элемента ввода текста с обеспечением защиты содержимого		
№	Атрибут	Функция
1	maxlength=*n*	Указание для данного элемента максимального количества вводимых символов
2	name=*строка*	Указание имени параметра, который передается в приложение обработки форм для данного элемента ввода. Этот атрибут является необходимым

Продолжение табл. 5

3	size=*n*	Указание для данного элемента количества отображаемых символов
4	value=*строка*	Задание исходного значения данного элемента
<input type=radio> – создание в **<form>** элемента ввода «селекторная кнопка»		
№	Атрибут	Функция
1	checked	Отображение элемента как выбранного по умолчанию
2	name=*строка*	Указание имени параметра, который будет передан в приложение обработки форм при выборе данного элемента ввода. Этот атрибут является необходимым
3	value=*строка*	Указание значения параметра, посылаемого в приложение обработки форм при выборе данного элемента. Этот атрибут является необходимым
<input type=reset> – создание в **<form>** кнопки сброса		
№	Атрибут	Функция
1	value=*строка*	Указание надписи для кнопки сброса
<input type=submit> – создание в **<form>** кнопки передачи		
№	Атрибут	Функция
1	name=*строка*	Указание имени параметра, который будет передан в приложение обработки форм при выборе данного элемента ввода. Этот атрибут является необходимым
2	value=*строка*	Указание надписи для кнопки передачи, а также значения параметра, посылаемого в приложение обработки форм для данного параметра при щелчке по этой кнопке
<input type=text> – создание в **<form>** элемента ввода текста (данный тип ввода используется по умолчанию)		
№	Атрибут	Функция
1	maxlength=*n*	Указание для данного элемента максимального количества вводимых символов
2	name=*строка*	Указание имени параметра, который передается в приложение обработки форм для данного элемента ввода. Этот атрибут является необходимым
3	size=*n*	Указание для данного элемента количества отображаемых символов
4	value=*строка*	Задание исходного значения данного элемента
<option>…**</option>** – начальный и конечный теги, с помощью которых определяется меню внутри элемента **<select>** в **<form>**		
№	Атрибут	Функция
1	selected	Отображение элемента как выбранного по умолчанию
2	value=*строка*	Возвращение указанного значения в приложение обработки форм вместо содержимого **<option>**
<select>…**</select>** – начальный и конечный теги, которые позволяют в **<form>** определить меню с несколькими вариантами выбора или список прокрутки с включением одного или более тегов **<option>**		
№	Атрибут	Функция
1	multiple	Разрешение выбора в **<select>** более одного значения **<option>**

2	name=*строка*	Определение текста, который будет передаваться в приложение обработки форм при выборе соответствующего значения **<option>**. Данный атрибут является необходимым
3	size=*n*	Отображение элементов меню будет организовано в виде ниспадающего меню, если size=1 (без указания атрибута *multiple*). В противном случае будет использоваться список прокрутки, включающий *n* элементов
<textarea>…</textarea> – начальный и конечный теги многострочной области ввода текста в форме. Содержимое тега **<textarea>** принимается как исходное значение по умолчанию		
№	Атрибут	Функция
1	cols=*n*	Задание отображения в текстовой области *n* столбцов текста
2	name=*строка*	Задание имени для значения текстовой области, которое передается в приложение обработки форм. Данный атрибут является необходимым
3	rows=*n*	Задание отображения в текстовой области *n* строк текста
4	wrap=*стиль*	Установка режима автоматического распределения текста в ячейке. Возможные значения: *off* (выключен); *virtual* (распределять текст по всей ячейке, но на сервер передавать как одну строку); *physical* (распределять текст по всей ячейке и передавать на сервер так, как он отображается)

Таблица 6. **Теги для создания бегущей строки**

<marquee>…**</marquee>** – начальный и конечный теги для создания бегущей строки		
№	Атрибут	Функция
1	align=*позиция*	Задание размещения бегущей строки по вертикали относительно текста, в котором она расположена: вверху (*top*), посередине (*middle*) или внизу (*bottom*)
2	behavior=*стиль*	Определение стиля бегущей строки как *scroll*, *slide* или *alternate*
3	bgcolor=*цвет*	Установка цвета фона для бегущей строки
4	direction=*направление*	Задание направления прокрутки текста: влево (*left*) или вправо (*right*)
5	height=*значение*	Задание высоты (в пикселах) зоны бегущей строки
6	hspace=*значение*	Задание размещения слева и справа от бегущей строки областей свободного пространства заданной ширины (в пикселах)
7	loop=*значение*	Установка числа повторов анимации бегущей строки. *Значение* может быть целым или *infinite*
8	scrollamount=*значение*	Задание размера (в пикселах) области смещения текста при каждой операции прокрутки

Окончание табл. 6

9	scrolldelay=*значение*	Указание задержки (в миллисекундах) между последовательными смещениями текста бегущей строки
10	vspace=*значение*	Задание размещения над бегущей строкой и под ней областей свободного пространства заданной высоты (в пикселах)
11	width=*значение*	Задание ширины зоны бегущей строки (в пикселах)

Приложение 2

Основные свойства CSS

Свойство	Возможные значения	Описание	Пример
1	2	3	4
font-family	[1] любая гарнитура шрифта	Определение используемого шрифта	font-family: Arial, Helvetica, Sans serif
font-size	[1] размер [2] xx-small, x-small, small, medium, large, x-large, xx-large – любое из этих значений [3] smaller, larger – любое из этих значений (относительный размер: меньше (больше), чем размер шрифта родительского элемента)	Размер шрифта	font-size: 30px
font-style	[1] normal – обычный [2] italic – курсив	Стиль шрифта: обычный или курсив	font-style: italic
font-variant	[1] normal – обычный [2] small-caps – малые прописные	Варианты отображения шрифта	font-variant: small-caps
font	[1] font-style [2] font-variant [3] font-weight [4] font-size [5] font-family	Сокращенная форма записи свойств шрифта	font: italic small-caps bold 12px Arial, Helvetica, sans-serif
font-weight	[1] любое значение от 100 до 900 с шагом 100	Насыщенность шрифта	font-weight: bold

Продолжение

1	2	3	4
font-weight	[2] normal – обычный (соответствует 400) [3] bold – полужирный (соответствует 700) [4] bolder – насыщенность выше, чем у шрифта родительского элемента [5] lighter – насыщенность ниже, чем у шрифта родительского элемента	Насыщенность шрифта	font-weight: bold
font	[1] font-style [2] font-variant [3] font-weight [4] font-size [5] font-family	Сокращенная форма записи свойств шрифта	font: italic small-caps bold 12px Arial, Helvetica, Sans serif
Текст			
text-align	[1] left – по левому краю [2] right – по правому краю [3] center – по центру [3] justify – по ширине	Горизонтальное выравнивание текста	text-align: center
text-indent	[1] длина [2] процент	Отступ первой строки текста (красная строка)	text-indent: 30px
line-height	[1] normal – обычный [2] длина [3] процент	Высота строки (междустрочный интервал)	line-height: 30px
vertical-align	[1] baseline [2] sub [3] super [4] text-top [5] top [6] middle [7] bottom [8] text-bottom	Вертикальное выравнивание внутри строчного блока	vertical-align: text-top
text-decoration	[1] none – обычный текст [2] underline – подчеркнутый	Оформление текста	text-decoration: underline

Продолжение

1	2	3	4
text-decoration	[3] overline – надчеркнутый [4] line-through – перечеркнутый [5] blink – мерцающий	Оформление текста	text-decoration: underline
text-transform	[1] none – текст выводится без изменений [2] capitalize – каждое слово начинается с прописной буквы [3] uppercase – все прописные [4] lowercase – все строчные	Изменение текста	text-transform: uppercase
letter-spacing	[1] normal – обычный [2] длина	Интервал между буквами	letter-spacing: 10px
Цвет и фон			
color	[1] цвет	Цвет текстового содержимого элемента	color: #F00
background-color	[1] цвет	Цвет фона элемента	background-color: #F00
background-image	[1] none – нет фонового изображения [2] URL графического файла	Фоновое изображение для элемента	background-image: url(ris.gif)
background-attachment	[1] scroll – фоновое изображение прокручивается вместе с содержимым документа [2] fixed – изображение не прокручивается, фиксируется в одном месте	Возможность прокрутки фонового изображения вместе с содержимым документа	background-attachment: fixed
background-repeat	[1] repeat – фоновое изображение повторяется по горизонтали и вертикали [2] repeat-x – фоновое изображение повторяется только по горизонтали	Повторение фонового изображения	background-repeat: repeat-x

Продолжение

1	2	3	4
background-repeat	[3] repeat-y – фоновое изображение повторяется только по вертикали [4] no-repeat – фоновое изображение не повторяется	Повторение фонового изображения	background-repeat: repeat-x
background-position	[1] расстояние от левого края + расстояние от верхнего края [2] left, center, right – одно из значений (выравнивание по горизонтали: по левому краю, по центру, по правому краю) [3] top, center, bottom – одно из значений (выравнивание по вертикали: по верхнему краю, по центру, по нижнему краю)	Положение фонового изображения	background-position: 50% 0% background-position: center top
background	[1] background-color [2] background-image [3] background-repeat [4] background-attachment [5] background-position	Сокращенная форма записи свойств фона	background: url(ris.gif) no-repeat fixed 50% 0%
Списки			
list-style-type	[1] disc, circle, square – одно из значений (для маркированного списка: закрашенный кружок, незакрашенный кружок, квадратик) [2] decimal, lower-roman, upper-roman, lower-alpha, upper-alpha – одно из значений (для нумерованного списка: арабские цифры,	Вид маркера для элементов списка	list-style-type: lower-alpha

Продолжение

1	2	3	4
list-style-type	строчные римские цифры, прописные римские цифры, строчные латинские буквы, прописные латинские буквы) [3] none – нет маркера	Вид маркера для элементов списка	list-style-type: lower-alpha
list-style-image	[1] none – нет маркера [2] URL графического файла	Изображение, которое будет использоваться в качестве маркера для элементов списка	list-style-image: url(ris.gif)
list-style-position	[1] outside – маркер располагается за пределами элементов списка (по умолчанию) [2] inside – при переносе следующие строки элемента списка будут располагаться под маркером без отступа	Положение маркера по отношению к элементам списка	list-style-position: inside
list-style	[1] list-style-type [2] list-style-position [3] list-style-image	Сокращенная форма записи свойств списка	list-style: square inside
Блочная модель			
width	[1] длина [2] процент	Ширина элемента	width: 50%
height	[1] длина [2] процент	Высота элемента	height: 100px
margin-top **margin-right** **margin-bottom** **margin-left**	[1] длина [2] процент [3] auto – автоматически	Наружный отступ сверху, справа, снизу и слева соответственно	margin-top: 100px margin-right: 10% margin-bottom: 10px margin-left: 100px
margin	[1] margin-top [2] margin-right [3] margin-bottom [4] margin-left	Сокращенная форма записи свойств отступов (можно указать одно значение, которое будет применяться ко всем четырем сторонам)	margin: 10px 20px 0px 10px

Продолжение

1	2	3	4
padding-top **padding-right** **padding-bottom** **padding-left**	[1] длина [2] процент	Внутреннее поле элемента (отступ от верхней/правой/нижней/левой границы элемента до содержимого)	padding-top: 100px padding-right: 10% padding-bottom: 5px padding-left: 100px
padding	[1] padding-top [2] padding-right [3] padding-bottom [4] padding-left	Сокращенная форма записи свойств внутренних полей	padding: 10px 20px 0px 10px
border-top-width **border-right-width** **border-bottom-width** **border-left-width**	[1] длина [2] thin, medium, thick – одно из значений (тонкая, средняя, толстая граница)	Ширина верхней, правой, нижней и левой границы элемента соответственно	border-top-width: thin border-right-width: medium border-bottom-width: 1em border-left-width: 2px
border-width	[1] border-top-width [2] border-right-width [3] border-bottom-width [4] border-left-width	Сокращенная форма записи свойств ширины границ элемента	border-width: 2px 1px 3px 1px
border-color	[1] цвет	Цвет границы элемента (можно указать одно значение, которое будет применяться ко всем четырем сторонам, или несколько значений для каждой стороны в отдельности)	border-color: green
border-style	[1] none – нет границы [2] dotted – линия из точек [3] dashed – пунктирная линия [4] solid – сплошная линия [5] double – двойная линия [6] groove – затемненная	Стиль границы элемента (можно указать одно значение, которое будет применяться ко всем четырем сторонам, или несколько значений для каждой стороны в отдельности)	border-style: dotted

Продолжение

1	2	3	4
border-style	[7] ridge – двухцветная [8] inset – двухцветная с затемнением снаружи [9] outset – двухцветная с затемнением внутри	Стиль границы элемента	border-style: dotted
border-top **border-right** **border-bottom** **border-left**	[1] border-width [2] border-style [3] border-color	Сокращенные формы записи свойств верхней, правой, нижней и левой границы элемента соответственно	border-top: 2px solid #000
border	[1] border-width [2] border-style [3] border-color	Сокращенная форма записи свойств всех границ элемента	border: thick double black
Позиционирование блоков			
position	[1] static – нормальный поток (по умолчанию) [2] relative – относительное позиционирование [3] absolute – абсолютное позиционирование	Задание модели позиционирования элемента (положение элемента определяется свойствами left, right, top и bottom)	position: absolute
left **right** **top** **bottom**	[1] длина [2] процент	Смещение блока от левого, правого, верхнего, и нижнего края контейнера соответственно (при относительном позиционировании элемент смещается относительно его местоположения в нормальном потоке, при абсолютном позиционировании смещение элемента определяется чаще всего относительно границ окна браузера)	left: 400px top: 100px

Окончание

1	2	3	4
float	[1] left – элемент всплывает к левому краю [2] right – элемент всплывает к правому краю [3] none – плавающая модель не применяется (по умолчанию)	Включение плавающей модели позиционирования элемента (всплытие элемента к левому или правому краю контейнера)	float: right
clear	[1] left – слева [2] right – справа [3] both – с двух сторон [4] none – по умолчанию	Определение расположения других элементов вокруг данного (указывает, какие стороны элемента не могут быть смежными со всплывшим элементом)	clear: left
z-index	[1] целые числа	Задание положения слоя по оси z, направленной перпендикулярно к экрану (определяет порядок наложения слоев)	z-index: 5

СОДЕРЖАНИЕ

Учебное издание

Шараева Ирина Викторовна
Прокопова Тамара Сергеевна
Ракутин Вячеслав Геннадьевич

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

ОСНОВЫ ВЕБ-ПРОЕКТИРОВАНИЯ

Курс лекций

Редактор *Н. Н. Пьянусова*
Технический редактор *Н. Л. Якубовская*
Корректор *С. Н. Кириленко*

Подписано в печать 29.09.2017. Формат 60×84 $^{1}/_{16}$. Бумага офсетная.
Ризография. Гарнитура «Таймс». Усл. печ. л. 3,49. Уч.-изд. л. 2,81.
Тираж 75 экз. Заказ .

УО «Белорусская государственная сельскохозяйственная академия».
Свидетельство о ГРИИРПИ № 1/52 от 09.10.2013.
Ул. Мичурина, 13, 213407, г. Горки.

Отпечатано в УО «Белорусская государственная сельскохозяйственная академия».
Ул. Мичурина, 5, 213407, г. Горки.