Частное учреждение образования
«Институт современных знаний имени А. М. Широкова»

Кафедра высшей математики и информатики

СОГЛАСОВАНО
Проректор по учебной и научной работе
Козлович М. И.

_____
23.02.2018 г.

ОСНОВЫ WEB-ДИЗАЙНА

Электронный учебно-методический комплекс
для студентов специальности 1-19 01 01 Дизайн (по направлениям),
направление специальности 1-19 01 01-06 Дизайн (виртуальной среды)

Составитель
Васильева Ю. Д., кандидат технических наук, доцент кафедры высшей математики и информатики частного учреждения образования «Институт современных знаний имени А. М. Широкова»;

Рассмотрено и утверждено
на заседании Совета Института
протокол № 8 от 27.03.2018 г.

УДК 004.92(078.5)
ББК 32.973я73

Р е ц е н з е н т ы:

кафедра «Системы автоматизированного проектирования» Учреждения образования Белорусский национальный технический университет (протокол № 8 от 26.02.2018 г.);

Гривачевский А. Г., заведующий лабораторией моделирования технологических процессов в Объединенном институте проблем информатики НАН Беларуси, доктор технических наук, доцент.

Рассмотрено и рекомендовано к утверждению
кафедрой высшей математики и информатики
(протокол № 7 от 01.02.2018 г.)

О75 **Васильева, Ю. Д.** Основы Web-дизайна : учеб.-метод. комплекс для студентов специальности 1-19 01 01 Дизайн (по направлениям), направление специальности 1-19 01 01-06 Дизайн (виртуальной среды) [Электронный ресурс] / Авт.-сост. Васильева Ю. Д. – Электрон. дан. (2,2 Мб). – Минск : Институт современных знаний имени А. М. Широкова, 2018. – 141 с. – 1 электрон. опт. диск (CD).

Систем. требования (миним.) : Intel Pentium (или аналогичный процессор других производителей) 1 ГГц ; 512 Мб оперативной памяти ; 500 Мб свободного дискового пространства ; привод DVD ; операционная система Microsoft Windows 2000 SP 4 / XP SP 2 / Vista (32 бит) или более поздние версии ; Adobe Reader 7.0 (или аналогичный продукт для чтения файлов формата pdf).

Номер гос. регистрации в НИРУП «Институт прикладных программных систем» 1201815224 от 14.04.2018 г.

Учебно-методический комплекс представляет собой совокупность учебно-методических материалов, способствующих эффективному формированию компетенций в рамках изучения дисциплины «Основы Web-дизайна».

Для студентов вузов.

ISBN 978-985-547-234-7

Пояснительная записка

Учебно-методический комплекс (УМК) по дисциплине «Основы Web-дизайна» разработан на основе учебной программы Института для студентов специальности 1-19 01 01 Дизайн (по направлениям) от 01 июля 2017 г., регистрационный номер № УД-02-185/уч. и предназначен для студентов дневной формы обучения.

УМК представляет собой совокупность учебно-методических материалов, способствующих эффективному формированию компетенций в рамках дисциплины «Основы Web-дизайна», изучается на протяжении двух семестров (со второго по третий курс).

УМК включает в себя курс лекций, планы лабораторный занятий, вопросы для самоподготовки к зачету и экзамену, учебную программу дисциплины, список литературы и ресурсов сети Интернет для освоения полного объема знаний, соответствующего стандартам высшей школы. Также приведен перечень программного обеспечения, используемого для практического освоения материала во время проведения лабораторных работ и методические рекомендации студентам по организации самостоятельной работы.

УМК поможет студентам получить базовые знания в области построения Web-страниц с учетом требований к уровню подготовки согласно образовательному стандарту Республики Беларусь, приобрести умения и навыки, без которых невозможно формирование компьютерной грамотности. Знания, умения и навыки, приобретенные в ходе изучения дисциплины, позволят студентам использовать современные программные средства для эффективного решения на высоком уровне специализированных задач, возникающих при дизайн-проектировании виртуальной среды

В учебно-методическом комплексе рассматриваются основные тенденции и требования, предъявляемых к дизайну Web-сайтов; раскрываются аспекты работы программной составляющей Web-сайта любой степени сложности; обеспечивается понимание основных принципов построения работы над Web-проектом.

1. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

1.1. Краткий курс лекций

ТЕМА 1. ОСНОВНЫЕ ПОНЯТИЯ WEB-ДИЗАЙНА

Структура интернет [2]

Сегодня трудно представить нашу жизнь без Интернета. Появившаяся в конце двадцатого века глобальная компьютерная сеть впитала в себя достижения человечества за многие века его существования и создала принципиально новые возможности общения людей. Современный интернет представляет собой сложнейшую систему из тысяч компьютерных сетей, объединенных, между собой. Состоит эта система из двух основных элементов – компьютеров (узлов) и соединяющих их коммуникационных каналов. Некоторая часть из этих компьютеров постоянно подключена к сети для оказания определенных услуг (сервисов) конечным пользователям. Такие компьютеры называются серверами. На серверах выполняются специальные программы – сервисы (службы). На каждом сервере работает одна или несколько различных служб, реализующих свои конкретные функции – прием или отправку электронной почты, передачи файлов, DNS и др.

Одна из самых популярных служб – WWW или Web-сервис. С помощью WWW сегодня за считанные минуты осуществляются различные операции – приобретение товаров и услуг, сложные финансовые транзакции, поиск нужных сведений, ознакомление с произведениями искусства и мультимедийными данными (фильмами, музыкой, фотографиями и телепередачами), доступ к глобальным библиотекам с огромными фондами самых разнообразных данных, общение в режиме реального времени с другими людьми и многое другое.

В основе WWW лежат разнообразные технологии обеспечивающие создание и доступ к гипертекстовым документам (HTML-страницам). Пользователи получают доступ к этим страницам, обращаясь к так называемым Web-узлам – WWW-сайтов.

Для того чтобы различные компьютерные системы могли эффективно взаимодействовать в сети интернет используются специальные протоколы.

Протоколами в компьютерных технологиях называют стандарты поведения, которых должны придерживаться различные компьютерные системы при взаимодействии друг с другом. Если провести аналогию с человеческим общением, то язык, на котором говорят люди, можно назвать протоколам общения. Чтобы понимать друг друга, людям необходимо использовать общепринятую лексику. То же самое и с компьютерной техникой: чтобы «общаться» друг с другом, два устройства должны использовать один и тот же общий протокол или, другими словами, «говорить на одном языке».

Основным протоколом интернет является протокол TCP/IP (приставка IP расшифровывается как Internet Protocol, протокол интернет). Он был утвержден в качестве официального стандарта для передачи данных в сети интернет в 1983 году. Этот протокол описывает порядок обмена данными между различными компьютерными системами (т.н. коммуникационный протокол) и представляет собой основание (корневую систему), на котором «вырастает» целый куст различных прикладных протоколов – электронной почты, передачи файлов, гипертекстовых сообщений, системы обмена сообщениями и др. Эти протоколы обеспечивают возможность реализации различных служб (интернет-сервисов). Исторически одним из первых интернет-сервисов является служба передачи файлов – FTP-сервис.

Одной из самых интересных возможностей интернета является доступ к гипертекстовым документам, находящимся на компьютерах сети. Гипертекст – это обычный текстовый документ, содержащий ссылки на другие документы, по которым можно переходить от одного документа к другому, но и возвращаться назад, к предыдущему тексту и т.д. Кроме того, гипертекстовые документы могут содержать иллюстрации, анимационные вставки, небольшие программные модули и т.д. Подавляющее большинство текстовой информации в интернете представлено именно гипертекстами. Технология гипертекста является основой для одной из самых значимых частей Интернета, называемой WWW (Всемирная паутина), WWW или просто Web.

Система доменных имен [2]

Интернет состоит из множества соединенных коммуникационными каналами компьютеров. В современном интернете используется большое количество различных протоколов, построенных на основе так называемого стека протоколов TCP/IP. Внизу этого стека находятся протоколы управления передачей, а вверху – протоколы взаимодействия с пользовательскими службами, например, такими, как электронная почта или передача файлов.

Непосредственную передачу данных от одного компьютера к другому обеспечивает протокол IP (Internet protocol). В соответствии с этим протоколом каждый компьютер в сети должен иметь уникальный цифровой адрес (4 байта для IP версии 4 и 16 байт для IP версии 6).

Людям трудно запоминать бессмысленные наборы цифр, поэтому для обозначения компьютеров они стали присваивать им мнемонические имена (например, www.google.com). Для того чтобы компьютеры могли определять по мнемоническим именам цифровые идентификаторы, был разработан специальный протокол и создана соответствующая служба, которая называется DNS (Dynamic Names Service) – служба динамических имен.

В соответствии с протоколом DNS весь интернет был поделен на группы, которые называются **доменными зонами**. Разделение по зонам может проводиться как по географическому, так и по тематическому признаку.

Географическая доменная зона определяет расположение компьютера в том или ином государстве. Вот несколько примеров географических доменов первого уровня; ru – Россия, fr – Франция, uk – Великобритания, jp – Япония, su – бывший Советский Союз, by – Беларусь.

Несмотря на обилие доменных зон, далеко не все из них пользуются большой популярностью. Основная часть компьютеров в интернете зарегистрирована в доменных зонах com и net.

Общее руководство и контроль над доменными зонами осуществляет организация ITANN (The Internet Corporation for Assigned Names and Number)– интернет-ассоциация по выдаче имен и чисел. Она передает полномочия на вы-

дачу адресов в той или иной доменной зоне другим организациям и следит за соблюдением основных правил.

Гипертекст [2]

Слово «гипертекст» было придумано Тедом Нельсоном (Ted Nelson). Впервые прозвучало в 1965 г. в его докладе «A File Structure for the Complex, the Changing, and the Indeterminate» (Файловая структура для сложных, меняющихся и неопределенных данных), прочитанном на национальной конференции в Нью-Йорке.

Классическое определение гипертекста Тед Нельсон сформулировал в 1987 г. Гипертекст – это «**форма письма, которое ветвится или осуществляется по запросу**». Иначе говоря, это «нелинейное письмо», которое «больше, чем текст». Обычно гипертекст представляется набором текстов, содержащих узлы перехода между ними, которые позволяют избирать читаемые сведения или последовательность чтения.

Для реализации возможности перехода между узлами используются гиперссылки. **Гиперссылкой** (англ. *hyperlink*) называется часть гипертекстового документа, указывающая на другой элемент, объект или на элементы этого объекта (команда, текст, заголовок, примечание, изображение). Элементы могут располагаться как в самом документе, так и в других объектах. Объектами могут быть файлы, каталоги, службы, расположенные на локальном диске или в компьютерной сети.

Гиперссылка может быть добавлена к любому элементу гипертекстового документа и обычно она выделяется графически.

Для обеспечения адресации Web-документов и других информационных объектов в свое время была предложена концепция **унифицированных указателей ресурса** (Uniform Resource Locator, URL). URL представляет собой набор информации, необходимый для того, чтобы определить:

- узел сети, на котором расположен информационный объект;
- расположение информационного объекта на узле;

- метод получения доступа к объекту.

Он может быть представлен в виде:

тип_ресурса://хост.домен:порт/имя_ресурса

Приведем примеры URL:

http://www.isz.by

mailto:name@isz.by

Как видно из примера, схема доступа может определять как конкретный протокол, так и целый сервис. В настоящее время существует целый ряд схем, которые могут использоваться в URL. Ниже перечислены наиболее часто использующиеся схемы:

Таблица 1. Тип ресурса

Схема	**Определяемый механизм доступа**
mailto	Электронная почта
news	Телеконференции
ftp	Протокол FTP
telnet	Протокол Telnet
http	Протокол HTTP
https	Протокол HTTPS
file	Файл, расположенный на локальном компьютере

Web-сервис. Просмотр гипертекстовых страниц [2]

В конце 1990 г. был создан первый Web-браузер, названный WWW, сначала для компьютеров NEXT, а затем и для других компьютерных платформ. С этого же момента начинается распространение Всемирной паутины по CERN и по всему интернету в целом. Всемирная паутина стала мощным стимулом для развития Интернета, позволяя представлять информацию в виде удобном для понимания и близком человеческому восприятию. Как признавались сами разработчики WWW, такое взрывообразное развитие их идей явилось для них самих полной неожиданностью.

Полные графики, удобные в использовании и красиво выглядящие HTML-документы, явились благодатной почвой для развития интернет-коммерции, в Сеть пошли потоки частных инвестиций, и появилась масса новых пользователей, которых интернет раньше отпугивал своей «академичностью» и непонятностью. Менее чем за 30 лет эксперимент, затеянный агентством ARPA, стал неотъемлемой частью человеческой культуры.

Важным этапом для коммерческого развития WWW стала разработка в 1994г. браузера Mosaic с открытым исходным кодом, давший «зеленый свет» разработке коммерческих Web-браузеров. На основе Mosaic, были впоследствии разработаны такие Web-браузеры, как Netscape Navigator и популярный Web-браузер Microsoft Internet Explorer.

В только начинающей свой путь Всемирной паутине царил некоторый хаос. Разные производители выпускали браузеры, частично не совместимые друг с другом, поскольку каждый стремился добавить в язык HTML свои теги форматирования, которые не поддержали конкуренты. В результате, часть HTML документов могли просматривать только одни браузеры, а часть – другие.

Недовольные этой неразберихой руководители CERN, совместно с Массачусетским Технологическим Институтом (MIT – Massachusetts Institute of Technology) сформировали в декабре 1994 года консорциум WWW (W3C), быстро взявший под свой контроль работу практически над всеми стандартами важнейших технологий Сети. Надо отметить, что формально W3C выпускает только рекомендации, и некоторые компании их игнорируют, но в целом рекомендации W3C признаются всем рынком в качестве стандартов.

Тема 2. Базовые составляющие Web-дизайна

Использование цвета в Web-дизайне [40]

В HTML цвет задается одним из двух путей: с помощью шестнадцатеричного кода и по названию некоторых цветов. Преимущественно используется способ, основанный на шестнадцатеричной системе исчисления, как наиболее универсальный.

Шестнадцатеричные цвета

Для задания цветов в HTML используются числа в шестнадцатеричном коде. Шестнадцатеричная система, в отличие от десятичной системы, базируется, как следует из ее названия, на числе 16. Цифры будут следующие: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Числа от 10 до 15 заменены латинскими буквами. В таблице 2 приведено соответствие десятичных и шестнадцатеричных чисел.

Таблица 2. Соответствие десятичных и шестнадцатеричных чисел до 15

Десятичные	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Шестнадцатеричные	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Числа больше 15 в шестнадцатеричной системе образуются объединением двух чисел в одно (таблица 3). Например, числу 255 в десятичной системе соответствует число FF в шестнадцатеричной.

Таблица 3. Соответствие десятичных и шестнадцатеричных чисел после 15

Десятичные	16	17	18	19	20	21	22	23	24	25	26	27	28
Шестнадцатеричные	10	11	12	13	14	15	16	17	18	19	1A	1B	1C

Чтобы не возникало путаницы в определении системы счисления, перед шестнадцатеричным числом ставится символ решетки #, например #aa69cc. При этом регистр значения не имеет, поэтому допустимо писать #F0F0F0 или #f0f0f0.

Типичный цвет, используемый в HTML, выглядит следующим образом.

```
<body bgcolor="#fa8e47">
```

Здесь цвет фона Web-страницы задан как #fa8e47. Символ решетки # перед числом означает, что оно шестнадцатеричное. Первые две цифры (fa) определяют красную составляющую цвета, цифры с третьей по четвертую (8e) — зеленую, а последние две цифры (47) — синюю. В итоге получится цвет в соответствии с рис. 1.

Рис. 1. Задание цвета в шестнадцатеричной системе

Каждый из трех цветов – красный, зеленый и синий – может принимать значения от 00 до FF, что в итоге образует 256 оттенков. Таким образом, общее количество цветов может быть 256x256x256 = 16.777.216 комбинаций. Цветовая модель, основанная на красной, зеленой и синей составляющей получила название *RGB* (red, green, blue; красный, зеленый, синий). Эта модель аддитивная (от add – складывать), при которой сложение всех трех компонент образует белый цвет.

Чтобы легче ориентироваться в шестнадцатеричных цветах существуют правила:

- Если значения компонент цвета одинаковы (например: #D6D6D6), то получится серый оттенок. Чем больше число, тем светлее цвет, значения при этом меняются от #000000 (черный) до #FFFFFF (белый).
- Ярко-красный цвет образуется, если красный компонент сделать максимальным (FF), а остальные компоненты обнулить. Цвет со значением #FF0000 самый красный из возможных красных оттенков. Аналогично обстоит с зеленым цветом (#00FF00) и синим (#0000FF).
- Желтый цвет (#FFFF00) получается смешением красного с зеленым.

Это хорошо видно на цветовом круге (рис. 2), где представлены основные цвета (красный, зеленый, синий) и комплементарные или дополнительные. К ним относятся желтый, голубой и фиолетовый (еще называемым пурпурным). Вообще, любой цвет можно получить смешением близлежащих к нему цветов. Так, голубой (#00FFFF) получается за счет объединения синего и зеленого цвета.

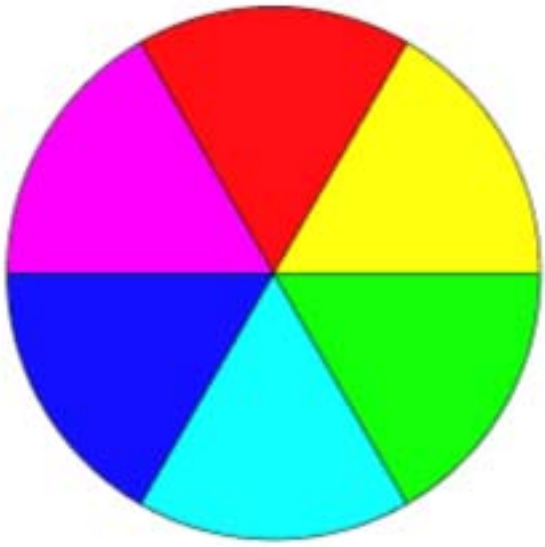


Рис. 2. Цветовой круг

Цвета по шестнадцатеричным значениям не обязательно подбирать эмпирическим путем. Для этой цели подойдет графический редактор, умеющий работать с разными цветовыми моделями, например, Adobe Photoshop. На рис. 3 показано окно для выбора цвета в этой программе, линией обведено полученное шестнадцатеричное значение текущего цвета. Его можно скопировать и вставить к себе в код.

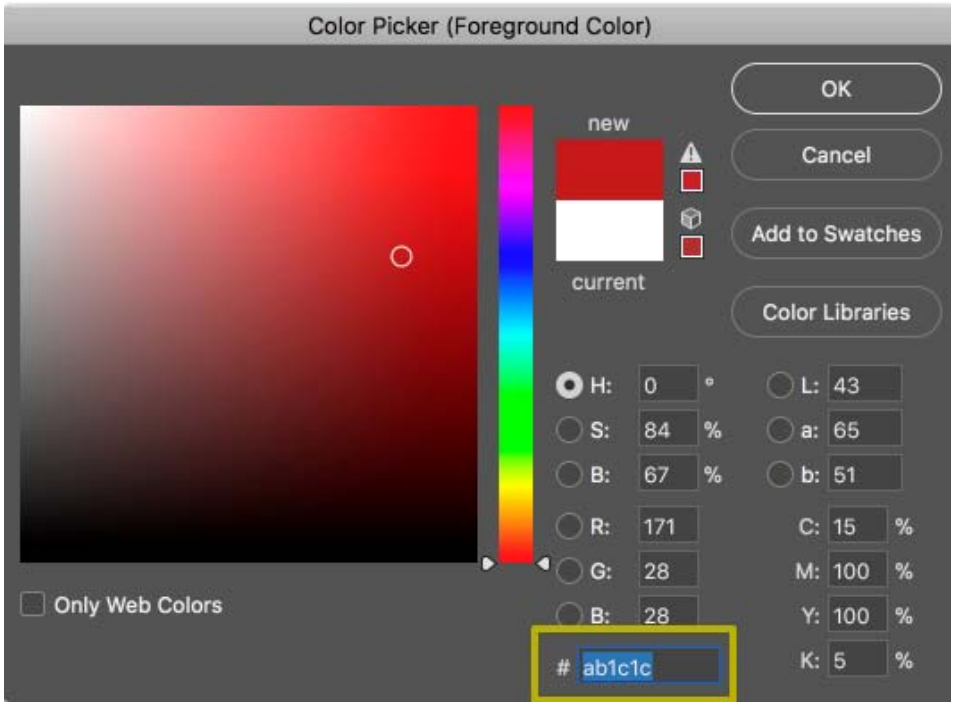


Рис. 3. Окно для выбора цвета в программе Photoshop

Web-цвета

Если установить качество цветопередачи монитора в 8 бит (256 цветов), то один и тот же цвет может показываться в разных браузерах по-своему. Это связано со способом отображения графики, когда браузер работает со своей собственной палитрой и не может показать цвет, который у него в палитре отсутствует. В этом случае цвет заменяется сочетанием пикселов других, близких к нему, цветов, имитирующих заданный. Чтобы цвет оставался неизменным в разных браузерах, ввели палитру так называемых Web-цветов. Web-цветами называются такие цвета, для каждой составляющей которых – красной, зеленой и синей – устанавливается одно из шести значений – 0 (00), 51 (33), 102 (66), 153 (99), 204 (CC), 255 (FF). В скобках указано шестнадцатеричное значение данной компоненты. Общее количество цветов из всех возможных сочетаний дает 6x6x6 – 216 цветов. Пример Web-цвета – #33FF66.

Основная особенность Web-цвета заключается в том, что он показывается одинаково во всех браузерах. В данный момент актуальность Web-цветов весьма мала из-за повышения качества мониторов и расширения их возможностей.

Цвет по названию

Чтобы не запоминать совокупность цифр, вместо них можно использовать имена широко используемых цветов. В табл. 4 приведены имена популярных названий цветов.

Таблица 4. Цветовые соответствия

Имя цвета	Цвет	Описание	Шестнадцатеричное значение
black		Черный	#000000
blue		Синий	#0000FF
fuchsia		Светло-фиолетовый	#FF00FF
gray		Темно-серый	#808080
green		Зеленый	#008000
lime		Светло-зеленый	#00FF00
maroon		Темно-красный	#800000
navy		Темно-синий	#000080

olive		Оливковый	#808000
purple		Темно-фиолетовый	#800080
red		Красный	#FF0000
silver		Светло-серый	#C0C0C0
teal		Сине-зеленый	#008080
white		Белый	#FFFFFF
yellow		Желтый	#FFFF00

Не имеет значения, каким способом вы задаете цвет – по его имени или с помощью шестнадцатеричных чисел. По своему действию эти способы равны. В следующем примере показано, как задать цвет фона и текста для страницы:

```
<body bgcolor="#fa8e47" text=white>
```

В данном примере цвет фона задается с помощью атрибута **bgcolor** тега **<body>**, а цвет текста через атрибут **text**. Для разнообразия значение у атрибута **bgcolor** установлено в виде шестнадцатеричного числа, а у **text** с помощью зарезервированного ключевого слова **white**.

Понятие формы в Web-дизайне

Для выполнения очень многих функций нужны на современных сайтах различные формы. С их помощью можно, например, общаться с посетителями (формы обратной связи), принимать заказы (формы онлайн-заказа), регистрировать новых пользователей (формы регистрации), оформлять подписку (формы подписки) и т. д. Разные формы для сайта имеют сегодня свои особенности создания и оформления [39].

Есть несколько типичных сценариев использования Web-форм [43]:

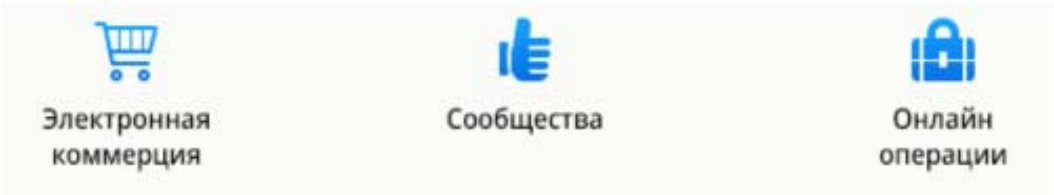
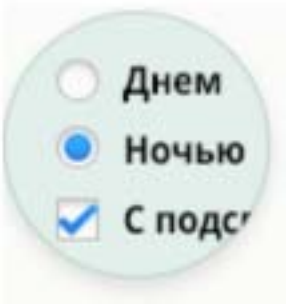


Рис. 4. Примеры использования Web-форм

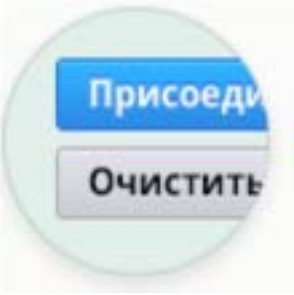
Количество необходимых полей в форме должно соответствовать задачам этой формы. Например, в форме заказа товаров на сайте поля для ввода полного имени, фамилии и отчества, а также контактного телефонного номера пользователя нужны обязательно – без этих данных невозможно будет подтвердить заказ. Разумеется, нужны на формах заказа товаров в большинстве случаев также поля для ввода адреса доставки товара, поля выбора способа оплаты товара и его доставки [39].

У большинства пользователей уже сложилось представление о том, как могут выглядеть формы, поэтому чаще всего они ожидают увидеть форму, состоящую из набора таких компонентов [43]:

Метки. Расположены рядом с формой ввода и подсказывают пользователю, что писать в соответствующем поле



Поля ввода и выбора. Большая группа элементов, которая включает в себя текстовые поля, кнопки переключения, элементы типа «чекбокс» и прочие.



Кнопки действия.Кнопки или ссылки, нажатие на которые вызывает определенное действие, например, отправку данных формы или очистку всех полей.

Сообщения. Любые текстовые сообщения, основанные на данных, которые ввел пользователь. Например, сообщение о неправильном адресе e-mail.

Подсказки. Текстовые блоки, более подробно описывающие поля. Могут содержать пример информации для заполнения, допустимые символы или ограничение по размеру загружаемого файла.

Проверки. Блоки, сообщающие пользователю о правильности вводимых им данных.

Чаще всего для описания метки достаточно одного-двух слов. В отдельных случаях можно использовать и более длинные подписи, если это необходимо для устранения неоднозначности.

Для полей ввода необходимо использовать соответствующие типы, к которым привыкли пользователи. Например, если в форме есть блок, в котором нужно выбрать только один вариант ответа – используйте кнопки переключения, если несколько – то кнопки типа «чекбокс».Если вам необходимо ограничить формат вводимых данных, например, создать поле с вводом даты, то это лучше сделать с помощь выпадающего списка или календаря, а не простого текстового поля ДД/ММ/ГГ.

Необходимо четко указывать пользователю на обязательные и необязательные для заполнения поля. Стандартным для этого символом является звездочка (* (asterisk)). На практике, символ может быть любым, если в форме присутствует легенда, объясняющая его предназначение.

Первичными действиями в форме могут считаться действия, завершающие процесс заполнения формы или перехода к следующему этапу, а вторичными – отмены, возврата или очистки данных.Одной из важных вещей в оформлении кнопок-действий является четкое визуальное разграничение по важности.

Сообщения об ошибках должны информировать пользователя, почему именно отправка формы закончилась неудачей. Выделяйте их красным фоном, границей, привычной иконографикой, другим начертанием или комбинируйте эти варианты.

Сообщения об успешных действиях пользователя должны сигнализировать пользователю, что он, к примеру, правильно заполнил один из разделов формы. Как и сообщения об ошибках, они должны сопровождаться понятной иконографикой, обращать на себя внимание, но не задерживать пользователя на его пути к цели.

Подсказки должны сопровождать поля, которым необходимы дополнительные пояснения. Например, чтобы объяснить пользователю, для чего будут использоваться его паспортные данные.

Описание не должно быть слишком длинным, и хорошей идеей было бы использование динамических подсказок, которые бы появлялись во время заполнения поля.

Текст в Web-дизайне [38]

Типографика – сильный инструмент для выражения посыла в Web-дизайне. С его помощью можно объединить текстовую и визуальную составляющие. Это свод законов, правил и норм оформления текста, основанных на изучении восприятия набора читателем. Знание и понимание типографики превращают текст в инструмент построения композиции, делают его живым, придают характер и способность передать идею не только при помощи содержания, но и графически.

Гарнитура – шрифт или несколько шрифтов, имеющих стилистическое единство начертания. Состоит из набора знаков. Часто это понятие путают с понятием «шрифт», хотя шрифт – это определенное начертание знаков, в то время как гарнитура определяет общее «семейство» шрифтов.

Гарнитуры можно разделить на две основные категории (рис. 5):

- Антиква – шрифты с засечками.

- Гротеск, соответственно, – шрифт без засечек.

В Web-проектах можно использовать любую гарнитуру. Нужно смотреть по ситуации, какой стиль, идея и что более актуально для него.

Рис. 5. Две группы гарнитур

Шрифт с засечками повышается удобочитаемость. Часто шрифты с засечками создают ощущение профессионализма и авторитетности предоставляемой информации, выражают уважение, подчеркивают стабильность и консерватизм в лучшем понимании этого слова. Шрифты без засечек, как правило, акцентируют рациональность, следование стилю, молодость и современность. Помогают создать пространство между буквами, а также отделить один знак от другого.

Кегль (рис. 6) – высота буквы, включающая в себя нижние и верхние выносные элементы. Измеряется в типографских пунктах (обозначается как pt). Например, текст, набранный 14 кеглем, будет равен 14 pt по высоте.

Интерлиньяж (рис. 6) – межстрочный интервал. Расстояние между базовыми линиями соседних строк.

Кернинг (рис. 6) – расстояние между буквами. Основная суть кернинга — подбор различных интервалов между различными парами конкретных букв для увеличения удобочитаемости.

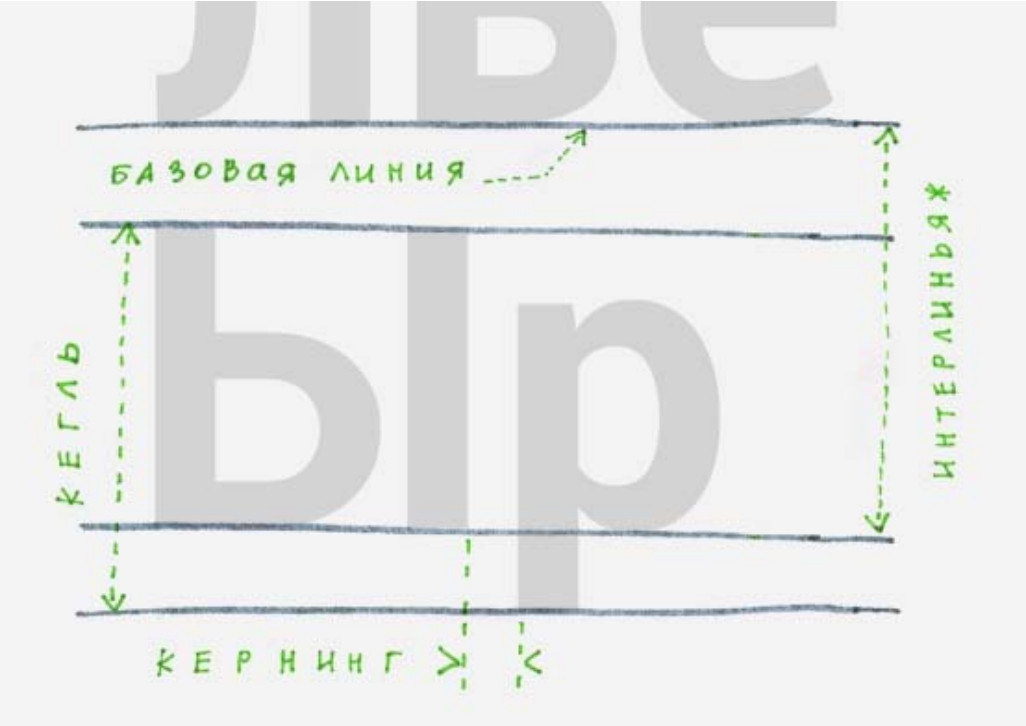


Рис. 6. Основные параметры шрифта

В Web-проекте желательно использовать не более 3-х начертаний. Это могут быть шрифты как одной гарнитуры, так и разных. Например, гарнитура Roboto содержит довольно большое количество различных начертаний. Из них легко можно выбрать три, которые подойдут для сайта. Допустим, это будут Light, Regular и Bold. Для заголовков можно использовать шрифт Bold или Light, для кнопок Bold, для основного текста Regular. Таким образом, используя одну гарнитуру, можно обеспечить сайт правильной типографикой. Естественно, все зависит от тематики сайта и идеи, которую планируется заложить в дизайн.

Размер текста в на Web-страницах не должен быть меньше 12 пикселей. Лучший выбор – в пределах 14–18 px для основного текста. Не слишком большой и в то же время удобочитаемый. Причем, если уж мы выбрали размер 16 px, он должен оставаться 16 px на всех страницах сайта и не скакать плюс-минус 1 px от блока к блоку. Относится это и к интерлиньяжу, он везде должен быть одинаков.

Размер шрифтов надо указывать целыми числами, не используя десятичных дробей, например 16,28 px.

Длина строки не должна превышать 600 px. Это оптимальный размер для комфортного перемещения взгляда с одной строчки на другую. Очень широкую контентную часть тяжело читать – часто теряется та строка, на которую собирался перейти после прочтения длинной предыдущей строки. Если все-таки необходимо растянуть текстовый блок на 1000 px и более по ширине, можно попробовать разбить текст на две или более колонок. Другой вариант — сделать межстрочное расстояние чуть больше обычного, чтобы визуально сильнее отделить строки друг от друга. Не забывайте разделять текст абзацами, это также поможет сделать его легко читаемым.

Расстояние между строками практически всегда должно быть больше размера шрифта. За исключением заголовков. Чтобы достигнуть баланса между текстом и «воздухом», можно сделать межстрочное расстояние примерно в полтора раза больше высоты строчных букв. Или устанавливается интерлиньяж, равный 150% размера шрифта. Например, размер текста 14 px, тогда интерлиньяж – 21 px. 14 / 2 = 7 + 14 = 21.

При разработке сайта почти никогда не используется тот контент, который там планируется быть – зачастую, это контента ещё просто не существует. Поэтому дизайнеры (да и верстальщики тоже) используют «рыбу» – произвольный текст, который вписывается в контентные блоки. Для текста используют генератор текста от компании Yandex.

Есть текстовые блоки, которые очень часто выравниваются неправильно – это выравнивание по ширине, выравнивание посередине и выравнивание по правому краю. Во всех этих случаях читать текст неудобно, и визуально он выглядит непривлекательно. Выравнивание ВСЕГДА должно быть по левому краю. Исключением может быть одно или два коротких предложения, которые, скорее всего, являются подзаголовками для основного текста.

Представление информации на одной странице

Лендинг (от англ. Landing page) – это всегда одностраничный сайт, который призывает к какому-либо одному действию. Лендинг, также называемый

«посадочная страница» или «страница захвата лидов», идеально подходит, чтобы донести четко обозначенную мысль до аудитории. В отличие от полноценного сайта, где много других разделов (о компании, продуктовые страницы, контакты, корзина и т. д.), лендинг удерживает внимание посетителей строго на одном сообщении и должен вызывать желание выполнить определенное целевое действие. Какое именно действие – зависит от ваших задач. Как правило, лендинг используется для сбора контактных данных потенциальных клиентов (они же – «лиды»), регистрации, перехода на страницу товаров или на главный сайт.

У одностраничного сайта есть своя структура и определенный стандартный набор элементов[41]:

1. **Заголовок**. Должен содержать максимально сжатую информацию, которая ясно выражает суть предложения Web-проекта. Это первое, что будет читать пользователь. Можно назвать это рекламным слоганом.

2. **Изображение**. Необходимо для привлечения внимания посетителя, для создания визуального положительного образа вашего продукта.

3. **Описание предложения**. Это может быть один абзац, или детальное описание, все зависит от желаемых целей. Главное – не увлекаться описанием товара. Ваша задача не вдаваться во все тонкости продукта, а склонить покупателя к действию различными приемами (будь то скидка, пробный продукт и т.д.)

4. **Отзывы**. Являются одним из обязательных элементов. Эта информация необходима для вызова доверия у потенциального клиента. Если посетитель хочет купить ваш продукт и видит, что другие клиенты уже воспользовались им и оказались довольны – повышается уровень его доверия, и возрастает вероятность того, что он купит.

5. **Иконки социальных сетей в подвале**. Они не должны быть ссылками на персональные профили в социальных сетях. Это должны быть профили страницы компании.

6. **Форма для заполнения информации**. Содержит поля для ввода информации, которую необходимо получить от клиента, для дальнейшей ее обработки и хранения в базе данных.

ТЕМА 3. СОЗДАНИЕ ГРАФИКИ ДЛЯ WEB-СТРАНИЦ

Форматы графических файлов [2]

Как известно, графическая информация в цифровых системах представляется двумя способами – векторным и растровым. При векторном представлении данных, графическая информация описывается в виде набора примитивов (круг, дуга, линя, прямоугольник) и связей между ними. При растровом способе – информация представляется в виде набора цветных точек (пикселей). Графические файлы служат для хранения изображений между сеансами работы с графическими программами и переноса изображений между программами и компьютерами. Графическая информация в файлах кодируется несколько иначе, чем в памяти компьютера. Более того, способов кодирования, называемых форматами, существует множество. Сосуществование большого числа форматов графических файлов обусловлено специфическими сферами их применения.

Для Web используются следующие основные форматы со сжатием информации:

- GIF – для хранения рисунков и анимаций (сжатие без потери качества).
- JPEG – для хранения фотографий (сжатие с потерей качества).
- PNG8 – для рисунков и фотографий в моделях Grayscale и Indexed (сжатие без потери качества).
- PNG24 – для рисунков и фотографий в модели RGB (сжатие без потери качества).

Самыми распространенными графическими форматами в Web являются *GIF* и *JPEG*. *GIF* — наиболее подходящий формат для обмена изображениями между системами. Архивы с изображениями в формате *GIF* можно найти на многих серверах в интернет. Данный формат поддерживают многие графиче-

ские приложения, в том числе все программы просмотра графики World Wide Web.

У формата GIF есть одно серьезное ограничение: он не поддерживает изображения с глубиной цвета больше восьми бит на пиксель. Обычно этого достаточно для контурных изображений типа комиксов и рисунков, где используется ограниченное количество цветов, или для небольших картинок, где для цветопередачи хватает 256 оттенков. Однако для больших изображений фотографического качества больше подходит формат *JPEG*.

Формат *GIF* использует один из лучших алгоритмов сжатия LZW, который изначально не предназначался специально для графики. Он не очень подходит для работы с двухцветными (черно-белыми) или фотографическими изображениями.

С развитием аппаратного обеспечения, поддерживающего высокое разрешение и богатую цветовую гамму, графические файлы значительно увеличились в размерах. Профессиональные художники теперь, как правило, работают с файлами, содержащими 10 и более мегабайт данных на каждое изображение. Даже пользователи с более скромными запросами подчас имеют дело с изображениями 640 на 480 пикселей в 256 цветах (а это более 300 килобайт). Кроме того, многие сейчас начинают работать с полноцветными изображениями 1024 на 768 пикселей (это более 2,3 мегабайт данных). Так как высококачественные изображения встречаются все чаще, ограничения, накладываемые традиционными методами сжатия (например, LZW), становятся все более ощутимыми.

Формат *JPEG* отличается от других графических форматов, прежде всего тем, что он использует метод сжатия с потерей информации. *JPEG* частично идентифицирует и удаляет ту информацию, которая несущественна для восприятия изображения. В результате *JPEG* может достигать высокого уровня сжатия без заметных потерь в качестве изображения.

Инструментарий Adobe Photoshop в контексте Web-разработок [33]

Программное обеспечение Adobe Photoshop очень часто используют как для создания печатного дизайна, так и для Web-дизайна различных сайтов. Эта программа дает возможность работать, используя разные разрешения и цветовые настройки.

Цветовые настройки, которые используются дизайнерами в разработке их проектов, довольно просты. Итак, существуют две самых популярных цветовых схемы: RGB (Red, Blue, Green) и CMYK (Cyan, Magenta, Yellow, Black). Цветовую схему RGB используют в целом для всей графики, отображаемой на экране, а CMYK используют при разработке печатного дизайна.

Создавая дизайн для Web-сайтов, пользуются цветовой схемой RGB. Существует очень много цветовых схем, которые взаимодействуют с палитрой цветов RGB. Их главная задача состоит в том, чтобы подобрать именно те цвета, которые присутствуют на мониторе или на каких-либо других экранах с RGB дисплеем.

Для дизайна Web-сайта более приемлемой считается цветовая схема sRGB, которая была разработана HP и корпорацией Microsoft. Пользуясь данной схемой, люди увидят в браузере очень близкие цвета к тем, которые используются при созданииdизайнаWeb-сайта в Adobe Photoshop. Зачастую именно цветовая палитра sRGB используется на большинстве компьютерных мониторов.

Для установки цветовой палитры sRGB необходимо перейти сначала в **Edit>Color Settings** (Редактировать > Настройки цвета). После выбирать цветовую палитру **North America General Purpose 2** (это надо сделать, если она не установлена по умолчанию). Изменив все настройки, нажмите на кнопку OK. Теперь необходимо удостовериться в том, что выбран цветовой профиль sRGB IEC61966-2.1.

Далее можно перейти к размерам и разрешением проекта. Размеры холста, на котором будет создан дизайн проекта, напрямую будет зависеть от самого проекта. Обычно стандартным размером Web-сайта принято считать ширину

в 1140 пикселей, но не помешает добавить еще по 150 пикселей с двух сторон. Это позволит увидеть задний фон страницы, дизайн которой создается.

Занимаясь созданием дизайна для Web-сайта, можно не обращать внимания на значение DPI/PPI изображения. В браузере он всегда имеет один и тот же вид. DPI/PPI могут отличаться только в том случае, если дизайн, который создается, должен пойти на печать. Форматы изображений PNG и GIF никогда не имеют настроек DPI/PPI. Главное в дизайне, что имеет значение, — это размер в пикселях.

С 2013 года компания Apple начала выпускать высокочастотные экраны Retina. Все изображения, которые создаются для таких типов экранов, должны иметь двойной размер. Это позволяет в дальнейшем избежать так называемого «замыливания» картинки. Поэтому следует сделать вывод, что создавая любой дизайн, необходимо разрабатывать два вида изображений. Одним из таких должно быть двойного размера специально для экранов Retina. Оно будет более четко отображаться в браузере.

Существует несколько способов создания дизайна в Photoshop, которые позволяют Web-дизайнеру быть уверенным в том, что его графика будет четко отображаться на дисплеях Retina. Первым таким способом является создание первоначального дизайна в 1x. В этом случае размер пикселей на холсте будет точно такой же, как и в браузере. Для этого необходимо использовать только векторные фигуры и изображения, которые должны с легкостью приобретать размеры, необходимые для экранов Retina, и не терять при этом свои графические качества.

Второй способ – это создание начального дизайна 2x. В данном случае размер пикселей должен быть в два раза больше, чем он будет отображаться в браузере. Следует обратить свое внимание на то, что все в дизайне должно быть в два раза больше. Используйте векторные фигуры и растровые изображения. Растровые изображения не поменяют своего качества после того, как они уменьшаться в размере. Если у вас дисплей Retina, то работая в Photoshop, все

будет четким, иметь резкость. А вот если работать за обычным монитором, то следует весь дизайн уменьшить до 50%.

Для того чтобы оптимизировать свой рабочий процесс необходимо правильно расположить инструменты в приложении. Все эти настройки можно назвать рабочим пространством. Программное обеспечение Adobe Photoshop позволяет создавать достаточно большое количество рабочих пространств и дает возможность переключатся на одно из них в любой нужный момент.

Рассмотрим окна, которые необходимы дизайнеру для работы:

Layers (Слои)**.** Эта панель используется намного чаще, чем другие, поэтому она должна быть крупных размеров и легкодоступна. Отдельные части дизайна сохраняются в виде отдельных слоев. Благодаря этому есть возможность корректировать каждый из них в отдельности.

Paragraph (Параграф). С помощью этой панели можно отредактировать весь параграф, задав ему нужную ширину текста и количество пробелов.

Info (Информация). Данное окно позволяет узнать любую информацию о палитре цветов и размерах.

Character (Символ). Здесь можно выбрать шрифт, задать его размер и т.д.

History (История). Любое движение или операция записывается в историю. Это позволяет в случае какой-либо ошибки отменить предыдущее действие или создать копию текущего состояния работы.

Navigator (Навигатор). Это окно обычно используют, чтобы быстро просмотреть дизайн.

Paragraph Styles (Стиль параграфа). Здесь можно создать типографические стили в виде заголовков и параграфов, что будет способствовать их повторному использованию.

Character Styles (Стиль символа). Эта панель поможет определить стили для ссылок и определенных слов.

Активизировать все эти окна можно в меню **Window**, кликая по ним. Также можно распределить все панели в произвольном порядке и сохранить их в качестве личного рабочего пространства. Воспользоваться своим рабочим ме-

стом можно будет когда угодно. Если одного места вам окажется мало, можно создать еще несколько. Это будет зависеть от размеров экрана, которым пользуетесь. На рис. 7 изображен пример того, как может выглядеть рабочие место на широкоформатном мониторе и на экране, размер которого составляет 15.6 дюймов.

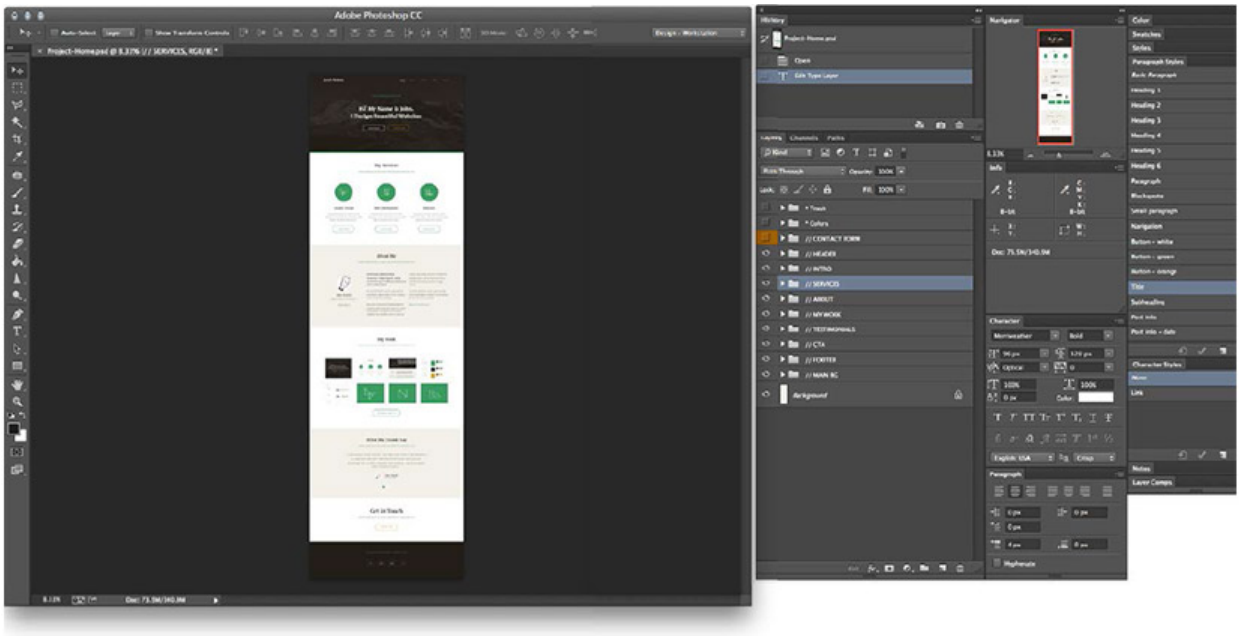


Рис. 7. Пример расположения окон

Тема 4. Стандарт HTML.

Основные концепции разметки

HTML-страницы[2]

Язык HTML позволяет размечать электронный документ, который отображается на экране с полиграфическим уровнем оформления; результирующий документ может содержать самые разнообразные метки, иллюстрации, аудио- и видеофрагменты и так далее. В состав языка вошли развитые средства для создания различных уровней заголовков, шрифтовых выделений, различные списки, таблицы и многое другое.

В качестве основы HTML был выбран обычный текстовый файл. Таким образом, гипертекстовая база данных в концепции WWW – это набор текстовых файлов, размеченных на языке HTML, который определяет форму представления информации (разметка) и структуру связей между этими файлами и

другими информационными ресурсами (гипертекстовые ссылки). Гипертекстовые ссылки, устанавливающие связи между текстовыми документами, постепенно стали объединять самые различные информационные ресурсы, в том числе звук и видео; в результате возникло новое понятие – гипермедиа.

Такой подход предполагает наличие еще одного компонента технологии – интерпретатора языка. В WWW функции интерпретатора разделены между Web-сервером гипертекстовой базы данных и интерфейсом пользователя. Сервер, кроме доступа к документам и обработки гипертекстовых ссылок, обеспечивает предпроцессорную обработку документов, в то время как интерфейс пользователя осуществляет интерпретацию конструкций языка, связанных с представлением информации.

В настоящее время WWW Consortium (W3C) – международная организация, которая занимается подготовкой и распространением документации на описание новых версий HTML – опубликовала материалы спецификации HTML5. Основная ее цель – улучшить язык, поддерживающий работу с новейшими мультимедийными приложениями, при этом сохраняется легкость чтения кода для человека и ясность исполнения для компьютеров и приспособлений (Web-браузеры, синтаксические анализаторы и т.д.). HTML5 включает в себя не только HTML 4, но и XHTML 1, а также DOM2HTML (особенно JavaScript). HTML5 отвечает всем требованиям, для которых HTML и XHTML в основном используются во всемирной паутине. HTML5 также считается микстурой будущего с введенными различными спецификациями, вместе с тем введенными продуктами программного обеспечения, такими, как Web-браузеры, установленными для использования в общей практике, а также исправления множества синтаксических ошибок, возникающих в существующих Web-документах. HTML5 – также попытка определить единый язык разметки, который мог бы быть написан как в HTML, так и в XHTML и был бы синтаксически корректен. Принятие стандарта HTML5 ожидается в 2012 г., но некоторые разработчики считают, что это произойдет гораздо позже.

HTML5 представляет несколько новых элементов и атрибутов, которые часто используются на современных Web-сайтах. Некоторые из них семантически заменены для общего использования базовых блоков и строковых элементов , например, <nav> (блок навигации по сайту), <footer> (обычно обращение к нижней части страницы или последней строке HTML кода), или <audio> и <video> вместо <object>. Некоторые элементы, которые можно было использовать в HTML 4.01, были исключены, например, представляемые элементы, такие как <font> и , чьи эффекты выполняются с помощью Cascading Style Sheets (Каскадные таблицы стилей). Также в поведении на Web-ресурсах снова заострено внимание на важности скриптов.

В предыдущие версии HTML, кроме возможностей разметки текста, включения мультимедиа и формирования гипертекстовых связей, включены дополнительные средства работы с мультимедиа, языки программирования, таблицы стилей, упрощенные средства печати изображений и документов. Для управления сценариями просмотра страниц Website (гипертекстовой базы данных, выполненной в технологии WWW) можно использовать языки программирования этих сценариев, например JavaScript и VBScript, а также Java.

Возможности интернет позволяют пользователям, владеющим основами HTML, создавать и размещать собственные Web-узлы без больших затрат.

Принципы гипертекстовой разметки [2]

HTML документ представляет собой набор элементов, каждый из которых может в свою очередь содержать другие элементы. Таким образом, элемент представляет собой набор контейнеров, в которые вложены другие контейнеры, и т.д.

Минимальным содержанием контейнера будут являться какие-то конкретные элементы. Такими элементами являются, например текст, который мы хотим отобразить на экране пользователя, изображения (рисунки), фильмы, музыка, анимация, а также вложенные HTML-документы и. т. д.

HTML является описательным языком разметки документов. Для описания элементов в нем используются указатели разметки (теги). Такая теговая модель описывает документ как совокупность контейнеров, каждый из которых начинается и заканчивается тегами, то есть документ HTML представляет собой не что иное, как обычный файл в кодировке ASCII с добавленными в него управляющими HTML-кодами (тегами). Поскольку HTML произошел от SGML, в нем разрешено использовать только три управляющих символа: горизонтальную табуляцию, перевод каретки и перевод строки. Это облегчает взаимодействие с различными операционными системами.

Тег – это управляющий HTML-код.

Теги HTML-документов в большинстве своем просты и понятны, ибо они образованы с помощью общеупотребительных слов английского языка, понятных сокращений и обозначений. HTML-тег состоит из *имени*, за которым может следовать необязательный *список атрибутов* тега. Текст тега заключается в угловые скобки ("**<**" и "**>**"). Простейший вариант тега – имя, заключенное в угловые скобки, например **<head>** или **<i>**. Для ряда тегов характерно наличие атрибутов, которые могут иметь конкретные значения, устанавливаемые автором для изменения функции тега.

Например, при описании таблицы открывающий тег с атрибутами может выглядеть так:

```
<table width=570 align=center cellpadding=10
       cellspacing=2 border=16>
```

Эта запись означает следующее: таблица шириной 570 пикселей выровнена по центру, поле между рамкой и содержимым ячеек – 10 пикселей, поле рамки – 2 пикселя, ширина бордюра – 16 пикселей.

Атрибуты тега следуют за именем и отделяются друг от друга одним или несколькими знаками табуляции, пробелами или символами возврата к началу строки. Порядок записи атрибутов в теге значения не имеет. Значение атрибута, если таковое имеется, следует за знаком равенства, стоящим после имени атри-

бута. Если значение атрибута – одно слово или число, то его можно просто указать после знака равенства, не выделяя дополнительно. Все остальные значения необходимо заключать в одинарные или двойные кавычки, особенно если они содержат несколько разделенных пробелами слов.

Чаще всего HTML-теги состоят из начального и конечного компонентов, между которыми размещаются текст и другие элементы документа. Имя конечного тега идентично имени начального, но перед именем конечного тега ставится косая черта (/) (например, для тега стиля шрифта – курсив ****закрывающая пара представляет собой****, для тега заголовка **<title>**закрывающей парой будет **</title>**). Конечные теги никогда не содержат атрибутов. Теги определяют область действия правил интерпретации текстовых документов.

При использовании вложенных тегов в документе следует соблюдать особую аккуратность. Вложенные теги нужно закрывать, начиная с последнего. Некоторые HTML-теги не имеют конечного компонента, поскольку являются автономными элементами. Например, тег изображения **<img>**, который служит для вставки в документ графического изображения, конечного компонента не требует. К автономным тегам также относятся разрыв строки (**
), горизонтальная линейка (<hr>**) и теги, содержащие такую информацию о документе, которая не влияет на его отображаемое содержимое, например теги **<meta>**и **<base>**.

Структура HTML-документа и элементы разметки заголовка документа [2]

Контейнер HTML или гипертекстовый документ состоит из двух других вложенных контейнеров: заголовка документа (**head**) и тела документа (**body**).

Основные контейнеры заголовка – это элементы HTML-разметки, которые наиболее часто встречаются в заголовке HTML-документа, т.е. внутри элемента разметки HEAD.

Рассмотрим только восемь элементов разметки, включая сам элемент разметки HEAD:

Head. Содержит заголовок HTML-документа. Служит для размещения информации, относящейся ко всему документу в целом. Данный элемент разметки не является обязательным. При наличии тега начала элемента разметки желательно использовать и тег конца элемента разметки. По умолчанию элемент **head** закрывается, если встречается либо тег начала контейнера **body**, либо тег начала контейнера **frameset**.

Title. Служит для именования документа в WWW. Более прозаическое его назначение – именование окна браузера, в котором просматривается документ. Состоит контейнер из тега начала, содержания и тега конца. Наличие тега конца обязательно. Тег начала элемента не имеет специфических атрибутов.

В различных браузерах алгоритм отображения элемента **title** может отличаться.

При выборе текста для содержания контейнера **title** следует учитывать, что отображается он системным шрифтом, так как является заголовком окна браузера. В нелокализованных версиях операционных систем и графических оболочек русский текст содержания элемента **title** будет отображаться абракадаброй.

Синтаксис контейнера **title**в общем виде выглядит следующим образом:

```
<title>Самый вкусный кофе - сайт о кофе</title>
```

Base. Служит для определения базового URL для гипертекстовых ссылок документа, заданных в неполной (частичной) форме. Кроме того, **base** позволяет определить мишень (окно) загрузки документа по умолчанию при выборе гипертекстовой ссылки текущего документа.

Meta. Это наиболее популярный элемент разметки заголовка, более распространен только элемент **title**. Такое положение дел объясняется назначением данного элемента разметки. **Meta** содержит управляющую информацию, кото-

рую браузер использует для правильного отображения и обработки содержания тела документа.

Можно построить автоматически перезагружаемую последовательность страниц. Для этого в заголовке каждой страницы из данной последовательности следует поместить соответствующий контейнер **meta**.

```
<meta http-equiv="refresh" content="1; url=refreshX.htm">
```

Кроме Content-type, можно указать и другие операторы. Например, запретить кэширование документа. Необходимость в этом возникает при частом обновлении документа или наличии в нем изменяющихся SSI- вставок.

С появлением роботов поисковых машин на **meta**-тег была возложена еще одна функция – описание поискового образа документа. Наиболее последовательно это было впервые реализовано в Webcrawler. До этого в качестве поискового образа документа использовался либо весь список слов документа, либо слова первого абзаца.

Собственно, для описания документа используется два **meta**-тега. Один определяет список ключевых слов, а второй – реферат (краткое содержание документа), который отображается в качестве пояснения к ссылке на документ в отчете поисковой машины о выполненном запросе. Контейнер **title** здесь также используется в качестве названия документа.

Link. Позволяет загружать внешние описатели стилей.

```
<link href="css/style.css" rel="stylesheet">
```

В данном случае речь идет о загрузке стилей из файла style.css. При этом стили задаются в нотации W3C, а не JavaScript, что определяется атрибутом type. В сущности, атрибут **rel** определяет тип гипертекстовой связи, **href** (Hypertext REFerence) указывает адрес документа, идентифицирующего связь, а атрибут **type** определяет тип содержания этого документа.

Style. Предназначен для размещения описателей стилей. При этом описание стиля из данного элемента разметки, если оно совпадает по имени класса и/или идентификатору подкласса со стилем, описанным во внешнем файле, заменяет описание стиля из внешнего файла. С точки зрения влияния на весь документ, описатели стилей задают правила отображения контейнеров HTML-документа для всей страницы.

Script. Служит для размещения кода JavaScript, VBScript или JScript. Вообще говоря, **script** можно использовать не только в заголовке документа, но и в его теле. В отличие от контейнера **style**, ему не требуется дополнительный контейнер **link** для загрузки внешних файлов кодов. Это можно сделать непосредственно в самом контейнере **script.**

Чаще всего применяются элементы **title**, **script**, **style**. Использование элемента **meta** говорит об осведомленности автора о правилах индексирования документов в поисковых системах и возможности управления HTTP-обменом данными. **Base** и **isindex** в последнее время практически не применяются. **Link** указывают только при использовании внешних относительно данного документа описателей стилей.

Заголовок HTML-документа является необязательным элементом разметки. Однако современная практика HTML-разметки такова, что почти в каждом документе есть HTML-заголовок.

Теги тела документа предназначены для управления отображением информации в программе интерфейса пользователя. Они описывают гипертекстовую структуру базы данных при помощи встроенных в текст контекстных гипертекстовых ссылок. Тело документа может состоять из комбинации элементов типа:

- иерархические контейнеры и заставки;
- заголовки (от H1 до H6);
- блоки (параграфы, списки, формы, таблицы, картинки и т.п.);
- горизонтальные подчеркивания и адреса;

- текст, разбитый на области действия стилей (подчеркивание, выделение, курсив);
- математические описания, графики и гипертекстовые ссылки.

Описание тегов тела документа следует начать с тега **body**. В отличие от тега HEAD, тег BODY имеет атрибуты (Таблица 5).

Таблица 5. Атрибуты контейнера ИЩВН

Атрибут	Значение
BACKGROUND = “path to image”	Источник фона
BGCOLOR=#XXXXXX	Цвет фона
TEXT=#XXXXXX	Цвет текста
VLINK =#XXXXXX	Цвет пройденных гипертекстовых ссылок
LINK =#XXXXXX	Цвет гипертекстовой ссылки

Пример HTML-страницы:

```
<html>
     <head>
    <meta charset="utf-8">
      <title>Основные возможности языка HTML</title>
      </head>

      <body>

      </body>
</html>

```

ТЕМА 5. ИСПОЛЬЗОВАНИЕ ТЕКСТОВЫХ И ГРАФИЧЕСКИХ ЭЛЕМЕНТОВ В HTML [2]

Теги управления текстом

Заголовок обозначает начало раздела документа и является основным тегом на Web-странице. В стандарте определено 6 уровней заголовков: от **h1** до **h6**. Текст, окруженный тегами **<h1></h1>**, получается большим – это основной

заголовок. Если текст окружен тегами **<h2></h2>**, то он выглядит несколько меньше (подзаголовок); текст внутри **<h3></h3>**еще меньше и так далее до **<h6></h6>**.

В реальности редко используется более трех уровней текста, а более 5 – крайне редко.

Для разделения текста на параграфы применяется тег <P>. В нем используются те же атрибуты, что и в заголовках.

Выравнивание текста осуществляется с помощью атрибута **align**, он позволяет выровнять текст по левому или правому краю, по центру или ширине. По умолчанию текст выравнивается по левому краю. Данный атрибут применим также к графике и таблицам.

Приведем возможные значения атрибута **align**:

- align=justify – выравнивание по левому и правому краям. Реализовано не во всех программах интерпретации;
- align=left – выравнивание по левому краю. По умолчанию текст HTML выравнивается по левому краю и не выравнивается по правому, то есть начало строк находится на одном уровне по вертикали, а концы – на разных. Чаще всего, полученный при этом текст с равными промежутками между словами выглядит лучше. Поскольку выравнивание по левому краю задается автоматически, атрибут align=left можно опустить;
- align=right – выравнивание по правому краю. Текст, выровненный по правому краю и не выровненный по левому – концы строк находятся на одном уровне, а начало на разных, – часто используется с целью создать оригинальный дизайн. Для этого задается атрибут **align**=right в обычных тегах, например в теге <P>;
- align=center – центрирование текста и графики. Есть несколько способов отцентрировать текст или графику. В спецификациях HTML 3.0 предлагается пользоваться тегом <align =center>. Однако этот тег применим не ко всем объектам HTML-страницы, поэтому разработчики Netscape добавили тег <**center**>, который центрирует любые объекты и поддерживается браузерами

Netscape Navigator 3.0, Microsoft Explorer 3.0 и другими. К тегу <**center**> нужно относиться с осторожностью. Какой-нибудь браузер может его вообще проигнорировать, и на странице окажется текст, выровненный по левому краю.

С помощью атрибута **align** можно заставить текст «обтекать» графический объект. Для этого следует поместить тег <img src="/путь/файл.gif"> туда, где должен быть графический объект, и добавить атрибут align=left, align=right или align=center.

Можно также создать рамку вокруг картинки или обрамление таблицы текстом. Чтобы текст не «обтекал» графику, а прерывался, необходимо применить тег <**br**> с атрибутом **clear**.

Принудительный перевод строки с помощью тега <**br**>используется для того, чтобы нарушить стандартный порядок отображения текста. При обычном режиме интерпретации программа интерфейса пользователя отображает текст в рабочем окне, автоматически разбивая его на строки. В этом режиме концы строк текста игнорируются. Иногда для большей выразительности требуется начать печать с новой строки. Для этого и нужен тег BR. Атрибут **clear** в теге <**br**> используется для того, чтобы остановить в указанной точке обтекание объекта текстом и затем продолжить текст в пустой области за объектом. Продолжающийся за объектом текст выравнивается в соответствии со значениями **left**, **right** или **all** атрибута **clear**:

```
<br clear=left> Текст будет продолжен, начиная с ближайшего
пустого левого поля.
<br clear=right> Текст будет продолжен, начиная с ближайшего
пустого правого поля.
<br clear=all> Текст будет продолжен, как только и левое, и
правое поля окажутся пустыми.
```

Для любого текста можно применить форматирования, изменяя его внешний вид и стиль. Существует набор тегов для управления отображением символов. Все эти теги можно разбить на два класса: 1) теги, управляющие формой отображения (font style); 2) теги, характеризующие тип информации

(information type). Часто внешне разные теги при отображении дают одинаковый результат. Это зависит главным образом от настроек интерпретирующей программы и вкусов пользователя.

К тегам, управляющие формой отображения относятся (таблица 6): курсив, усиление, подчеркивание, верхний индекс, нижний индекс, шрифт большой, маленький, красный, синий, различные комбинации. Microsoft Internet Explorer и Netscape Navigator позволяют определить шрифт с помощью атрибута FONT. Теперь можно объединять на одной странице несколько видов шрифтов, вне зависимости от того, какой из них задан по умолчанию в браузере пользователя.

Таблица 6. Теги, управляющие формой отображения

Тег	**Значение**
<I>...</I>	Курсив (Italic)
<B>...</B>	Усиление (Bold)
<TT>...</TT>	Телетайп
<U>...</U>	Подчеркивание
<S>...</S>	Перечеркнутый текст
<BIG>...</BIG>	Увеличенный размер шрифта
<SMALL>...</SMALL>	Уменьшенный размер шрифта
_{...}	Подстрочные символы
^{...}	Надстрочные символы

Теги для управления формой отображения допускают вложенность и пересечение друг с другом, поэтому все они имеют тег начала и конца. При использовании таких тегов следует помнить, что их отображение зависит от настроек программы-интерфейса пользователя, которые могут и не совпадать с настройками программы-разработчика гипертекста.

Теги, характеризующие тип информации, приведены в таблице 7.

Таблица 7. Теги, характеризующие тип информации

Тег	Значение
<EM>...</EM>	Типографское усиление
<CITE>...</CITE>	Цитирование
<STRONG >...</STRONG>	Усиление
<CODE >...</CODE>	Отображает примеры кода (например, "коды программ")
<SAMP>...</SAMP>	Последовательность литералов
<KBD>...</KBD>	Пример ввода символов с клавиатуры
<VAR>...</VAR>	Переменная
<DFN>...</DFN>	Определение
<Q>...</Q>	Текст, заключенный в скобки

Блоки цитат

Для создания блоков цитат используется тег <**blockquote**>. Тег добавляет поля слева и справа от текста. Это полезный тег, поскольку он позволяет компактно расположить текст в центре страницы.

При неоднократном использовании <**blockquote**> текст все больше сжимается к центру.

Создание списков в HTML

Списки являются важным средством структурирования текста и применяются во всех языках разметки. В HTML имеются следующие виды списков: неупорядоченный список (ненумерованный) (Unordered Lists <**ul**>), упорядоченный список (нумерованный) (Ordered Lists <OL>) и список определений. Теги для неупорядоченных и упорядоченных списков – это основа HTML. HTML 3.2 добавляет несколько атрибутов к тегам списков для выбора разных типов маркеров в неупорядоченных списках и разных схем нумерации в упорядоченных. Можно включать такие атрибуты и в сами теги элементов списка (List Item <**li**>), чтобы сменить тип маркера в середине списка. После появления нового атрибута все последующие маркеры в списке будут иметь такой же вид.

Неупорядоченный список предназначен для создания текста типа:

- первый элемент списка;

- второй элемент списка;
- третий элемент списка.

Записывается данный список в виде последовательности:

```
<ul>
    <li>первый элемент списка</li>
    <li>второй элемент списка</li>
    <li>третий элемент списка</li>
</ul>
```

Теги <**ul**> и </**ul**> – это теги начала и конца неупорядоченного списка, тег <**li**> (List Item) задает тег элемента списка. Помимо этих тегов, существует тег, позволяющий именовать списки – <**lh**> (List Header).

Упорядоченный список задается с помощью тега <**ol**>. В качестве номеров не только обычные числа, но и строчные и прописные буквы, а также строчные и прописные римские цифры. При необходимости можно даже смешивать эти типы нумерации в одном списке

```
<ol>
    <li>Чебурашка</li>
    <li>Крокодил Гена</li>
    <li>Шапокляк</li>
</ol>
```

Горизонтальные линейки

Горизонтальное подчеркивание (Horizontal Rule) применяется для разделения документа на части. С помощью одного лишь тега <HR> можно придать странице оригинальный вид. Попробуйте поэкспериментировать с тегом <**hr**>, и вы получите линии, совсем не похожие на те, которыми обычно пользуетесь.

Комментарии в языке HTML

При разметке документов HTML возникает необходимость в использовании комментариев, которые браузер не выводит на экран, но другой специа-

лист, редактирующий данный документ, может прочитать. В таких примечаниях можно найти информацию о том, кто является автором документа, где и почему используется конкретный элемент HTML и т.п. Комментарии HTML начинаются с символа "<!--" и оканчиваются символом -->". Можно вставлять текст с любыми символами. Комментарии могут состоять из нескольких строк текста. В общем и целом они ничем не отличаются от аналогичных комментариев в других языках программирования, так как видимы только тогда, когда это необходимо. Например, браузер игнорирует их.

При создании файла HTML можно разместить в нем комментарии о его структуре. Кроме того, там можно размещать информацию о том, какие сложные операции способен выполнять данный документ.

Гипертекстовые ссылки в HTML-документе

Все рассмотренные выше средства управления отображением текста, безусловно, важны, но они только дополняют основной тег HTML-документа– гипертекстовую ссылку. Для записи гипертекстовой ссылки используется тег <A>, который называют «якорь» (anchor). Якорь имеет несколько атрибутов, главным из которых является HREF. Простую ссылку можно записать в виде <ahref="http://www.isz.minsk.by"> Отображаемое название гипертекстовой ссылки, где значение атрибута href – адрес документа "index.htm" на машине "www.intuit.ru", доступ к которой осуществляется по протоколу HTTP. Форма записи этого адреса называется универсальным локатором ресурсов URL и является составной частью технологии WWW.

По умолчанию в качестве базового используется URL каталога, в котором находится текущий документ. Если URL начинается с символа "." или "..", то это означает исчисление от текущего каталога. Если URL начинается с символа "/", то относительный URL берется от корня каталогов сервера.

Одна из особенностей создания Web-сервера состоит в том, что представленную на нем информацию желательно разбить на отдельные части, которые могут быть выведены на экран без необходимости его прокрутки. Организация

связей между отдельными частями осуществляется с помощью гипертекстовых ссылок.

```
<a href="http://www.isz.by/history">Наш институт</a>
```

При нажатии на ссылку в окно браузера будет загружен новый документ.

Другой формой использования тега <A> является определение точек внутри текста, на которые можно сослаться. Такой метод применяется в том случае, когда документ нельзя поделить на части и необходимо быстро перемещаться из оглавления в текст:

```
<a href="http://www.isz.by/#point">Ссылка на
точку "point" в документе "index.html"</a>


<a name=point>Точка перехода</a>
```

Вставка изображений в Web-страницы

Для того чтобы вставить в Web-страницу изображение, необходимо либо нарисовать его, либо взять уже готовое. В любой программе рисования можно создать простое изображение и сохранить его в нужном формате. Если программа этот формат не поддерживает, необходимо преобразовать файл в требуемый формат. Существует множество программ, предназначенных для преобразования одного графического формата в другой. Позаимствовать же *картинки* можно из различных программных пакетов или с других Web-страниц в интернете, содержащих библиотеки свободного доступа художественных изображений. Когда браузер выводит Web-страницу с изображением, соответствующий графический файл временно хранится в памяти компьютера. В большинстве браузеров есть команда, позволяющая сохранить файл на локальном диске. Существует также множество других вариантов получения графических файлов.

Изображения могут содержать определенную информацию, да и просто придают Web-странице привлекательный вид. Приведем наиболее распространенные случаи применения изображений:

- логотип компании на деловой странице;
- графика для рекламного объявления;
- различные рисунки;
- диаграммы и графики;
- художественные шрифты;
- подпись автора страницы;
- применение графической строки в качестве горизонтальной разделительной линии;
- применение графических маркеров для создания красивых маркированных списков.

Рассмотрим, как вставить изображение в Web-страницу. Тегом HTML, который заставляет браузер выводить изображение, является <IMG> с обязательным атрибутом **src** (SouRCe, источник). Имя файла представляет собой имя выводимого графического файла. Замыкающего тега не требуется.

Пример вставки изображения:

```
<img src="image.gif" alt="изображение">
```

Изображения на Web-странице могут использоваться в качестве гипертекстовых ссылок, как и обычный текст. Пользователь щелкает на изображении и отправляется на другую страницу или переходит к другому изображению. Для обозначения изображения как гипертекстовой метки используется тот же тег **<a>**, что и для текста, но между <a> и вставляется тег изображения **<img>**:

```
<a href="http://isz.by"><img src="image.gif"
alt="изображение"></a>
```

При этом изображение, используемое в качестве гипертекстовой ссылки, обводится дополнительной рамкой.

Тег изображения имеет один обязательный атрибут SRC и необязательные: **alt, align, usemap, hspace, vspace, border, width, height**.

Атрибут src. Указывает файл изображения и путь к нему; изображение должно быть загружено в браузер и размещено в том месте документа, где расположен тег изображения.

Атрибут alt. Позволяет указать текст, который будет выводиться вместо изображения браузерами, неспособными представлять графику. В некоторых случаях при недостаточной пропускной способности линий связи пользователи отключают отображение графики. Наличие названий вместо картинок облегчает восприятие Web-страниц в таком режиме.

Атрибут align. Определяет положение изображения относительно окружающего его текста. Возможные значения аргумента – ["top" | "middle" | "bottom"] ("вверху", "посередине", "внизу").

Align="top" выравнивает верх изображения по верхнему краю самого высокого элемента в строке окружающего текста.

Align="middle" выравнивает центр изображения по базовой линии строки окружающего текста.

Align="bottom" выравнивает нижний край изображения по базовой линии строки окружающего текста.

Кроме основных значений атрибута ALIGN="ключевое слово" существует еще ряд аргументов, которые расширяют возможности взаимного размещения графики и текста. Рассмотрим их подробнее.

Дополнительные возможные значения аргумента – **["left" | "right" | "top" | "texttop" | "middle" |"absmiddle" | "baseline" | "bottom" | "absbottom"]**.

Align="left" определяет огибаемое текстом изображение. Изображение располагается вдоль левой границы документа, а последующие строки текста огибают его справа.

Align="right" определяет огибаемое текстом изображение. Изображение располагается вдоль правой границы документа, а последующие строки текста огибают его слева.

Align="top" выравнивает верх изображения по верхнему краю самого высокого элемента в строке окружающего текста точно так же, как при использовании стандартного набора атрибутов.

Align="texttop" выравнивает верх изображения по верхнему краю самого высокого текстового символа в строке окружающего текста. Действие этого аргумента в большинстве случаев, но не всегда, подобно действию аргумента ALIGN="top"**.**

Align="middle" выравнивает центр изображения по базовой линии строки окружающего текста точно так же, как при использовании стандартного набора атрибутов.

Align="absmiddle" выравнивает центр изображения по центру строки окружающего текста.

Align="baseline" выравнивает нижний край изображения по базовой линии строки окружающего текста, то есть производит такое же действие, как и ALIGN="bottom".

Align="bottom" выравнивает нижний край изображения по базовой линии строки окружающего текста точно так же, как при использовании стандартного набора атрибутов.

Align="absbottom" выравнивает нижний край изображения по нижнему краю строки окружающего текста.

Атрибут usemap. Если присутствуют атрибут **usemap** и тег **<map>**, изображение становится чувствительной картой, или "графическим меню". Если щелкнуть кнопкой мыши на активной области изображения, для которого определен атрибут **usemap**, произойдет гипертекстовый переход к информационному ресурсу, установленному для этой области. Более подробно этот вопрос будет рассматриваться в следующем разделе.

Атрибут border. Целочисленное значение аргумента определяет толщину рамки вокруг изображения. Если значение равно нулю, рамка отсутствует. Чтобы не вводить пользователей в заблуждение, не стоит задействовать BORDER=0 в изображениях, которые представляют собой часть элемента якоря, поскольку рисунки, применяемые в качестве гиперссылок, обычно выделяются цветной рамкой.

Атрибут hspace. Целочисленное значение этого атрибута задает горизонтальное расстояние между вертикальной границей страницы и изображением, а также между изображением и огибающим его текстом.

Атрибут vspace. Целочисленное значение этого атрибута задает вертикальное расстояние между строками текста и изображением.

Атрибуты width и height. Оба атрибута задают целочисленные значения размеров изображения по горизонтали и по вертикали соответственно. Это позволяет уменьшить время загрузки страницы с графикой. Браузер сразу отводит рамку для изображения и продолжает загружать текст на страницу. Пока загружается графика, пользователь может начать читать текст. Определить размер изображения нетрудно, для этого достаточно воспользоваться любой программой просмотра графических файлов, например ACDSee или графическим редактором Corel PhotoPaint или Adobe Photoshop. Откройте файл в графическом редакторе и определите размер *картинки* в пикселах. В теге изображения следует указать ширину и высоту *картинки*.

```
<img src="image.gif" alt="изображение"
width="100" height="200" hspace="10" vspace="10"
border="2" align="left">
```

Создание таблиц

Тег **<table>** создает таблицу. Все прочие элементы таблицы должны быть вложенными в него. Допускается также вложение таблиц одна в другую, т.е. содержимым ячейки может быть другая таблица. Закрывающий тег обязателен.

Атрибуты:

align – выравнивание таблицы относительно документа. Возможные значения: center (по центру), right (по правому краю);

background – строка, определяющая рисунок для заднего фона;

bgcolor – определяет задний фон таблицы;

border – толщина рамки в пикселях. Если атрибут не указан, то таблица выводится без видимой рамки;

bordercolor – цвет рамки;

cellspacing – задает расстояние между ячейками таблицы;

cellpadding – задает расстояние между содержимым ячейки и ее рамкой

rules – описывает рамки вокруг таблицы. Может принимать следующие значения (таблица 8).

Таблица 8. Значения атрибута rules для задания рамки вокруг таблицы

all	Отображает все части рамки внутри таблицы
cols	Отображает все вертикальные рамки внутри таблицы
groups	Отображает горизонтальные части рамки между группами таблицы
none	Удаляет все рамки вокруг таблицы
rows	Отображает все горизонтальные рамки внутри таблицы

summary – описание таблицы для удобства людей, использующих браузеры, поддерживающие азбуку Бройля или речевой вывод;

title – всплывающая подсказка;

width – ширина таблицы в процентах или пикселях.

Тег **<tr>** (table row, строка таблицы) создает строку таблицы. Весь текст, другие теги и атрибуты, которые требуется поместить в одну строку, должны размещаться между тегами <**tr**></**tr**>.

Атрибуты:

align – Выравнивает текст в ячейке. Возможные значения: left (по левому краю), rigth (по правому краю), center (по центру).

valign – Выравнивает текст в ячейке по вертикали. Возможные значения: top (по верхнему краю). middle (по центру), bottom (по нижнему краю).

Тег **<td>**определяет отдельную ячейку в таблице.

Атрибуты:

height – указывает высоту элемента в процентах или пикселях;

align – выравнивает текст в ячейке, left – по левому краю (по умолчанию), rigth – по правому краю, center – по центру;

valign – выравнивает текст в ячейке по вертикали, top – по верхнему краю, middle – по центру, bottom – по нижнему краю;

colspan – указывает количество столбцов, которое объединено в одной ячейке (по умолчанию=1);

rowspan – указывает количество строк которое объединено в одной ячейке (по умолчанию=1);

title – всплывающая подсказка.

Тег **<th>** определяет заголовок для столбца в таблице. Обычно выделяются жирным шрифтом. Закрывающий тег обязателен.

Атрибуты:

bgcolor – цвет фона;

bordercolor – цвет рамки для элемента;

height – указывает высоту элемента в процентах или пикселях;

align – выравнивает текст в ячейке, left – по левому краю (по умолчанию), rigth – по правому краю, center – по центру;

valign – выравнивает текст в ячейке по вертикали, top – по верхнему краю, middle – по центру, bottom – по нижнему краю;

colspan – указывает кол-во столбцов, которое объединено в одной ячейке (по умолчанию=1);

rowspan – указывает кол-во строк, которое объединено в одной ячейке (по умолчанию=1);

title – всплывающая подсказка.

Таблица 9. Пример таблицы в HTML

<table>
<tr><td colspan="3">Заголовок таблицы</td></tr>
<tr><td>Заголовок столбца 1</td><td>Заголовок столбца 2</td><td>Заголовок столбца 3</td></tr>
<tr><td colspan="3">Если в таблице два тега TR, то в ней две строки.</td></tr>
<tr><td>Если в строке три тега TD,</td><td>то в ней</td><td>три столбца.</td></tr>
</table>

Верстка документов с помощью таблиц

Таблицы изначально проектировались как средство представления двумерных массивов данных, но сейчас широко используются в основном как инструмент для точного позиционирования контента на Web-странице.

При верстке с помощью таблиц следует придерживаться некоторых принципов и избегать распространенных ошибок.

Принципы табличной верстки:

- Любой горизонтальный блок следует реализовывать в виде отдельной таблицы.
- Уровень вложенности таблиц – не более 3.
- Чем меньше строк и колонок, тем лучше.
- Чем меньше изображений и чем меньше их объем, тем лучше.

Распространенные ошибки:

- Забывают прописать **cellspacing** и **cellpadding**, что приводит к появлению разрывов между картинками.
- Забывают задать **border**="0" для рисунков, которые являются ссылками, из-за чего вокруг них появляются рамки.
- Не указывают фоновый цвет документа для элемента <**body**>.
- Используют абсолютные ссылки с указанием домена.

- Ссылку на главную страницу сайта можно сделать так: Главная, а можно вот так: Главная.
- Используют прописные буквы в именах файлов и папок. (Например, файл называют LOGO.gif, а в коде страницы пишут <img src="images/logo.gif ">).
- Используют тег <**font**>. (Если вдруг понадобится изменить размер или гарнитуру шрифта на всем сайте, то придется последовательно и аккуратно просматривать многие страницы, заменяя атрибуты тега <**font**>).
- Забывают указать ширину и высоту рисунков. Если сайт построен на базе таблиц, то это значительно увеличивает время загрузки его страниц.
- Забывают указать **valign**="**top**" для таблицы.
- Делают ширину страницы больше 760 пикселей. Если сайт построен на базе таблиц с фиксированной шириной, то у посетителей с разрешением экрана 800×600 появится горизонтальная полоса прокрутки. Она затрудняет навигацию по сайту и страшно раздражает пользователей.
- Организуют большое количество вложенных таблиц. Практически все макеты можно сверстать, используя таблицы двух уровней вложенности, и только для некоторых требуется три уровня.
- Забывают прописать атрибут **alt** для рисунков. Во-первых, некоторые посетители отключают отображение картинок; во-вторых, правильный альтернативный текст делает страницу более адаптированной к поисковым системам.
- Создают плохой заголовок. Элемент <**title**> очень важен для поисковых систем. В современном интернете поисковые системы имеют огромное значение и приводят на сайт около 50% посетителей. Правильный заголовок должен кратко и точно описывать содержание страницы.
- Отсутствует мета-информация. Как и заголовок страницы, мета- информация важна для поисковых систем. Необходимо прописать ключевые слова и краткое содержание страницы.

ТЕМА 6. ИНТЕРАКТИВНЫЕ ЭЛЕМЕНТЫ В HTML

Задание форм[2]

Формы были созданы и используются в WWW для получения отклика пользователя на предоставленную информацию и сбора данных о пользователе. После заполнения пользователем формы и запуска процесса ее обработки информация из нее попадает к программе, работающей на сервере. Простота использования тега <**mailto**:>в формах позволяет даже владельцам небольших страниц получать отклик от своих пользователей. Для обработки большого количества откликов используются программы, поддерживающие Common Gateway Interface (CGI) и расположенные на сервере, в адрес которого поступают отклики. Таким образом пользователь может интерактивно взаимодействовать с Web-сервером через Интернет.

Элемент **form** обозначает документ как форму и определяет границы использования других тегов, размещаемых в форме. Тег **<form>** определяется последовательностью тегов **<input>**, размещенных внутри пары <**form**> и </FORM>. В форме используется как метод (method), так и действие (action) для описания обработки данных, вводимых пользователем в форму. Метод (GET или POST) определяет, как должны обрабатываться входные данные из формы, а действие указывает на URI (Uniform Resource Identifier) программы, ответственной за обработку этих данных.

```
<form method=post
action=mailto:yourname@your.email>

</form>
```

Определение элементов управления формы

Для определения области внутри формы, куда вводятся данные, используется тег <**input**>. Он формирует поле для ввода информации пользователем. Это может быть текстовое поле, опция, изображение или кнопка. Вид поля ввода определяется значением атрибута **type**.

Когда пользователю необходимо ввести небольшое количество текста (одну или несколько строк), используется тег <**input**>, и атрибут **type** устанавливается в значение **text**. Это значение принято по умолчанию и указывать его необязательно. Кроме того, задается атрибут **name** для определения наименования переменной поля.

```
<form method=post>

    <input name="Name" size="35">

</form>
```

Имеется еще три дополнительных атрибута, которые можно использовать. Первый называется **maxlength**, он ограничивает число символов, вводимых пользователем в текущее поле. По умолчанию данное число не ограничено. Вторым атрибутом является **size**, определяющий размер видимой на экране области, занимаемой текущим полем. Значение по умолчанию определяется типом браузера. Если значение **maxlength** больше, чем **size**, браузер будет прокручивать данные в окне. Последним из дополнительных атрибутов является атрибут **value**, обеспечивающий начальное значение поля ввода.

Для создания независимых флагов в формах HTML используется тег <**input**> с атрибутом **type=checkbox**. В зависимости от содержания формы пользователь может отметить несколько флагов. Когда форма использует тег <**input**> с атрибутом **checkbox**, в нем должны присутствовать также атрибуты **name**, и **value**. Атрибут **name** указывает на наименование данного поля (флага) ввода, а в атрибуте **value** будет содержаться значение поля.

```
Россия<input name="Страна" type="checkbox"
value="Россия">
<br>
Страны СНГ<input name="Страна" type="checkbox"
value="Страны СНГ">
```

В некоторых случаях необходимо инициализировать данный флаг как уже отмеченный. В таких случаях тег <**input**>должен содержать атрибут **checked.**

В некоторых случаях требуется организовать выбор одного из нескольких возможных значений. Для создания формы ввода при выборе пользователем одного значения из нескольких возможных необходимо использовать тег **<input>** с атрибутом **type=radio**. Когда в форме применяется данный атрибут, в теге **<input>** должны быть указаны атрибуты **name** и **value**. Атрибут **name** указывает наименование соответствующего поля (кнопки). Атрибут **value** содержит значение поля.

```
Пол мужской<input name="Пол" type="radio"
value="Мужской">
<br>
Пол женский<input name="Пол" type="radio"
value="Женский">
```

Если в форме требуется организовать ввод пароля, то атрибут **type** можно установить в значение **password** (**type=password**). Используя данный тип, можно организовать ввод пароля без вывода на экран составляющих его символов. При этом следует помнить, что введенные данные передаются по незащищенным каналам связи и могут быть перехвачены.

```
Логин<input name="Логин" type="text">
<br>
Пароль<input name="Слово" type="password">
```

Когда пользователь заполняет форму, ему может потребоваться начать все сначала. На такой случай существует кнопка **Reset**, по которой пользователь может щелкнуть мышью, чтобы вернуться к первоначальным значениям полей. Когда пользователь выбирает данную кнопку, форма восстанавливает первоначальные значения всех элементов, в которых присутствует атрибут **type=reset**. Для создания кнопки **Reset** используется тег <**input**> с атрибутом **type=reset**. Браузер в свою очередь будет выводить изображение данной кнопки. Если в форме используется атрибут **reset**, тег <**input**> может дополнительно содержать атрибут **value**. Данный атрибут определяет надпись на изображении кнопки.

```
<input type="reset" value="Очистить форму">
```

Используя форму HTML для ввода информации от пользователя, необходимо обеспечить пользователю возможность завершить ввод данных. Для этого используется тег <**input**> с атрибутом **type=submit**. Браузер, в свою очередь, выводит данный элемент как кнопку, по которой пользователь может щелкнуть, чтобы завершить процесс редактирования. Когда в форме используется тег <**input**> с атрибутом **submit**, данный элемент может содержать два дополнительных атрибута: **name** и **value**. Атрибут **name** хранит название переменной поля в вашей форме. Атрибут **value** – определяет значение элемента формы, которое будет отправлено на сервер или получено с помощью клиентских скриптов.

```
<input type="submit" value="Отправить сообщение">
```

Скрытые поля. Добавление в тег **input** атрибута **type=hidden** позволит включить в отправляемую форму значения атрибутов **name** и **value**, которые пользователь изменить не может. Такие метки полезны при наличии нескольких форм для дальнейшей обработки данных.

Создание многострочных областей ввода текста

В зависимости от типа формы может потребоваться организовать ввод большого количества текста. Для создания текстового поля из нескольких строк используется тег <**textarea**>. Данный тег использует три атрибута: **cols**, **name** и **rows**.

Атрибут **cols**. Указывает (число символов) число колонок, содержащихся в текстовой области.

Атрибут **name**. Определяет наименование поля.

Атрибут **rows**. Задает количество видимых строк текстовой области.

```
 <textarea name="тема" cols="38" rows="3">

</textarea>
```

Использование списков в форме

Когда формы HTML становятся более сложными, в них часто включают списки с прокруткой и выпадающие меню. Для этого используют тег **select** с атрибутом **type=select**. Для определения списка пунктов используют тег **<option>**. Тег **<select>** поддерживает три необязательных атрибута: **multiple**, **name** и **size**.

Атрибут **multiple**. Позволяет выбрать более чем одно наименование.

Атрибут **name**. Определяет наименование объекта.

Атрибут **size**. Определяет число видимых пользователю пунктов списка. Если в форме установлено значение атрибута **size**=1, то браузер выводит на экран список в виде выпадающего меню. В случае **size** > 1 браузер представляет на экране обычный список.

В форме может использоваться тег <**option**> только внутри тега **<select>**. Эти теги поддерживают два дополнительных атрибута: **selected** и **value**.

Атрибут **selected**. Используется для первоначального выбора значения элемента по умолчанию.

Атрибут **value**. Указывает на значение, возвращаемое формой после выбора пользователем данного пункта. По умолчанию значение поля равно значению тега **<option>** (рис.8).

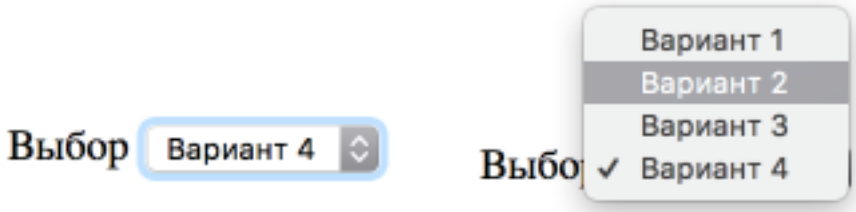


Рис. 8. Пример использования свитков

Методы форм

Для передачи заполненных форм на сервер применяются POST и GET.

Метод GET. Это простой запрос файла, каким пользуется браузер, когда вы набираете адрес в адресной строке или щелкаете по ссылке. При использовании этого метода браузер добавляет данные формы, оформленные как **переменная=значение** и разделенные знаком «&», непосредственно к имени файла,

указанного в атрибуте **action**, после знака «?». Таким способом может работать любая ссылка, оформленная соответствующим образом, например:

```
<a href="http://myhost.by/file.htm?param1=number&param2=12345">
Тестовая ссылка с параметрами</a>
```

Достоинство метода заключается в его универсальности и неприхотливости: Посредством данного метода можно обращаться к любому файлу.

У данного метода имеется три недостатка:

1. Кэширование данных, запрашиваемых данным методом серверами и браузерами. Зачастую это только ускоряет процесс загрузки, но иногда может привести к неверному отображению данных. Например, если в справочнике телефонов вы запрашиваете телефоны всех людей по фамилии «Иванов», то, поскольку результирующий список не меняется каждую секунду, выдача списка из кэша ускорит отображение страницы.

2. Ограничение длины запроса 256 символами является вторым недостатком метода GET. С учетом того, что в эту строку запроса должен вместиться еще и путь к запрашиваемому файлу, начиная с «http://», на передачу параметров остается мало места и его может не хватать для больших форм. На передачу файлов, места не хватит в любом случае.

3. Отображение запроса в адресной строке так же можно отнести к недостаткам метода. Не все люди понимают, что происходит в адресной строке и, соответственно, могут неправильно на нее реагировать.

Метод POST. Этот метод вместе с запросом файла, указанного в атрибуте ACTION, посылает дополнительный запрос, не отображаемый в адресной строке. Данные, полученные этим методом, не кэшируются, а длина запроса не ограниченна, вследствие чего появляется возможность отправлять с данными формы целые файлы. В этом заключается основные достоинства метода. Однако у него имеется 2 недостатка:

1. Данным методом можно обращаться только к CGI или PHP- скриптам. Обычно такие файлы имеют расширения «.php3», «.php», «.cgi», «.pl» и «.py» и

не индексируются поисковыми роботами. Исключение страниц из поисковых индексов может быть недопустимым.

2. Если на странице с формой, использующей данный метод, посетитель нажал на кнопку «Обновить», браузер сообщит о невозможности обновить страницу без повторной отправки данных и спросит разрешения отправить данные повторно. Если посетитель подтвердит повторную отправку данных, эффект может быть из ряда вон выходящим. Например, на странице регистрации пользователя повторная отправка данных, скорее всего, приведет не к отображению текста вроде «Спасибо за регистрацию», а к появлению сообщения, что такой пользователь уже существует (он был зарегистрирован при первой отправке формы). В результате, посетитель, не понимая, что зарегистрировался, может попытаться зарегистрироваться под другим именем (и не дай бог ему придет в голову еще раз нажать на кнопку «Обновить»). В итоге, база зарегистрированных пользователей будет содержать большее количество пользователей, чем их окажется на самом деле, что вряд ли соответствует пожеланию владельца регистрационной формы. Посетителю такое поведение тоже неудобно – ведь данные приходится вводить по нескольку раз. Если же на запрос о повторной отправке ответить отрицательно, браузер, вместо требуемой страницы, выдаст бланк с надписью о том, что страница устарела и к ней нет доступа, а так же, предложит нажать на кнопку «Обновить» – и тогда ситуация повторится.

В связи со своими особенностями, каждый из методов лучше применять в соответствии с ситуацией. Однако даже тогда, когда метод является наиболее подходящим к ситуации, его недостатки метода могут привести к невозможности его применения, если не позаботиться об их устранении.

Работа с фреймами

Фреймы позволяют выводить в одном окне браузера одновременно несколько Web-страниц, имеющих разные URL, различные полосы прокрутки (они могут и отсутствовать) и ведущих себя довольно независимо. На рис. 9 показана страница, интерфейс которой включает фреймы.

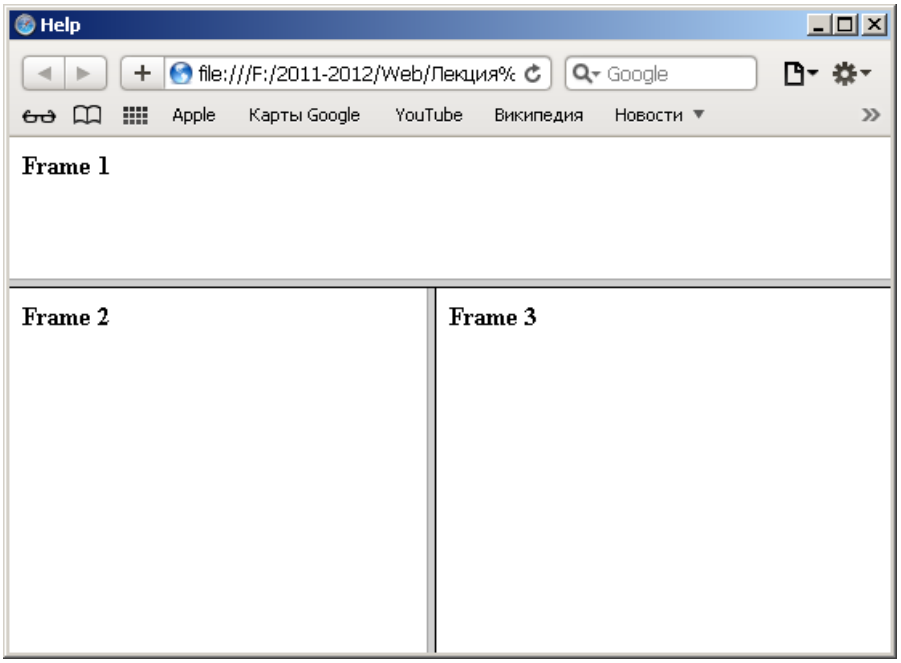


Рис. 9. Окно с фреймами

В каком-то смысле фрейм– это именно то, что означает данное слово: рамка вокруг картинки, окошко или страница. Вводя тег <**frame**>, дизайнер HTML-страницы разделяет экран браузера на части. В результате человек, просматривающий страницу, может изучать только одну ее часть, независимо от остального содержимого. Фактически браузер, распознающий фреймы, загружает разные страницы в разные секции, или фреймы экрана. Например, вы можете построить страницу таким образом, что фирменный знак будет зафиксирован в верхней части экрана, в то время как остальную часть страницы пользователь пролистывает обычным способом. Можно расположить сбоку кнопки навигации, которые не перемещаются, когда читатель щелкает по ним мышкой, так что изменяется только часть экрана, а сама полоска навигации остается неподвижной.

Построим страницу с двумя фреймами. Зададим слева фрейм оглавления с заголовками статей, а справа поместим страницу с самими статьями. Сделаем следующим образом: когда пользователь щелкает мышкой на ссылке в той части экрана, где находится оглавление, сама статья появляется в правом фрейме. Это основной, наиболее распространенный способ использования фреймов.

Задание фреймовой структуры

Для начала мы должны представить себе общий вид страницы – где расположить фреймы, и какого они будут размера (рис. 10). Затем можно подумать об их содержании. Ниже приводится код простой фреймовой структуры с использованием тега **<frameset>**. Обратите внимание: страница с фреймовой структурой не содержит тега **<body>**.

```
<html>
 <head>
  <meta charset="utf-8">
  <title>Пример фрймов</title>
 </head>

 <frameset cols="25%,75%">
  <frame src="menu.html">
  <frame src="main.html" name="main">
 </frameset>

</html>
```

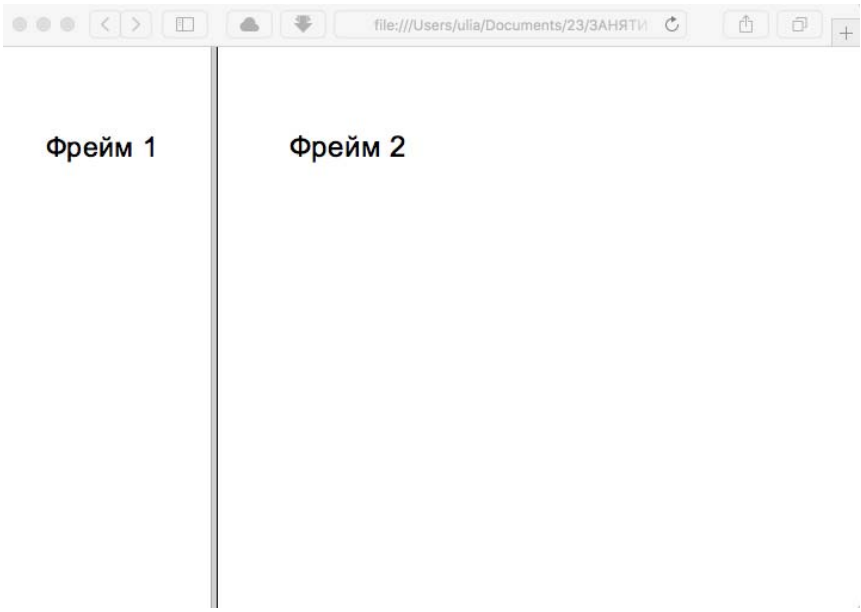


Рис. 10. Пример расположения фреймов

В результате мы получили экран, разделенный на два окна. Левое окно занимает 25% экрана и содержит страницу с названием menu.html. Окно справа займет 75% экрана и содержит файл main.html. Пока у нас их нет, так что вы

увидите страницу с двумя пустыми фреймами. Прежде чем она появится, нам придется пару раз щелкнуть мышкой в ответ на сообщения об ошибках, потому что браузер будет пытаться найти несуществующие страницы. Заметьте, что правую страницу мы назвали «main» (**<главная>**) с помощью строки:

```
<frame src="main.html" name="main">
```

Это означает, что фрейм под именем **main** будет содержать страницу **main.html**. Заметим, что поскольку мы не собираемся показывать в левом фрейме никаких страниц, кроме menu.html, нам не нужно его называть.

Теперь загрузим фреймы с содержимым. Зададим страницу menu.html в левом фрейме, где мы собираемся щелкать мышью, переключаясь между двумя страницами в правом фрейме. menu.html – это обычная HTML-страница, построенная как оглавление. На самом деле мы можем взять готовую страницу с оглавлением и использовать ее. Из-за того, что данный фрейм узкий и высокий, то страница, которая будет в него загружаться, должна быть спроектирована соответствующим образом. Теперь мы должны определить, где будут появляться другие страницы при щелчке мышкой на ссылке. Поскольку мы хотим, чтобы они отображались в правом фрейме, добавим атрибут **target** (**target**="**main**") в тег ссылки. Это означает, что, когда пользователь щелкает на ссылке, вызываемая страница появляется во фрейме **main**. Мы отображаем все страницы в фрейме **main**, поэтому нужно добавить атрибут **target**="**main**" во все теги ссылок в оглавлении. Если мы не определим атрибут **target**, то страница появится там, где мы щелкнули мышкой, — в левом фрейме.

Правый фрейм **main** будет содержать сами HTML-страницы. Ваша задача – спроектировать их так, чтобы они хорошо смотрелись в меньшем, чем обычно, окне, потому что часть экрана будет занята левым кадром оглавления. Но больше эти страницы ничем не примечательны.

Теги <**frameset**> обрамляют текст, описывающий компоновку фреймов. Здесь размещается информация о числе фреймов, их размерах и ориентации (горизонтальной или вертикальной). У тега <**frameset**> только два возможных атрибута: **rows**, задающий число строк, и **cols**, задающий число столбцов. Меж-

ду тегами <**frameset**> не требуется указывать тег <**body**>, но его можно поместить между тегами <**noframes**> в конце фреймовой структуры. Между тегами <**frameset**> не должно быть никаких тегов или атрибутов, которые обычно используются между тегами <**body**>. Единственными тегами, которые могут находиться между тегами <**frameset**>и </**frameset**>, являются теги <**frame**>, <**frameset**> и <**noframes**>. Это упрощает задачу. В основном все связано с тегами <**frame**> и их атрибутами. Если же вы хотите поэкспериментировать, можно создать вложенные друг в друга теги <**frameset**> аналогично тегам <**table**>.

Для каждой строки и столбца, упомянутых в теге <**frameset**>, необходим свой набор тегов <**frame**>.

Атрибут **rows** тега **<frameset>** задает число и размер строк на странице. Количество тегов <**frame**> должно соответствовать указанному числу строк. Справа от знака "=" можно определить размер каждой строки в пикселях, процентах от высоты экрана или в относительных величинах (обычно это указание занять оставшуюся часть места). Следует пользоваться кавычками и запятыми, а также оставлять пробелы между значениями атрибутов. Например, следующая запись формирует экран, состоящий из трех строк: высота верхней – 20 пикселей, средней – 80 пикселей, нижней – 20 пикселей:

```
<frameset rows="20,80,20">
```

Следующий тег – <**frameset**> – создает экран, на котором верхняя строка занимает 10% высоты экрана, средняя – 60%, а нижняя – оставшиеся 30%:

```
<frameset rows="10%,60%,30%">
```

Можно задать относительные значения в комбинации с фиксированными, выраженными в процентах или пикселях. Например, следующий тег создает экран, на котором верхняя строка имеет высоту 20 пикселей, средняя – 80 пикселей, а нижняя занимает все оставшееся место:

```
<frameset rows="20,80,*">
```

Столбцы **cols** задаются так же, как строки. Для них применимы те же атрибуты.

Тег <**frame**> определяет внешний вид и поведение фрейма. Этот тег не имеет закрывающего тега, поскольку в нем ничего не содержится. Вся суть тега <**frame**> в его атрибутах. Их шесть: **name**, **marginwidth**, **marginheight**, **scrolling**, **noresize** и **src**.

Атрибут **name**. Если вы хотите, чтобы при щелчке мышью на ссылке соответствующая страница отображалась в определенном фрейме, необходимо указать этот фрейм, чтобы страница "знала", что куда загружать. В предыдущих примерах мы назвали большой правый фрейм **main**, и именно в нем появлялись страницы, выбранные из оглавления в левом фрейме. Фрейм, в котором отображаются страницы, называется целевым (**target**). Фреймы, которые не являются целевыми, именовать не обязательно. Например, можно записать такую строку:

```
<frame src="my.html" name="main">
```

Имена целевых фреймов должны начинаться с буквы или цифры. Одни и те же имена разрешается использовать в нескольких фреймовых структурах. По щелчку мыши соответствующие страницы будут отображаться в именованном фрейме.

Атрибут **marginwidth**. Действует аналогично атрибуту таблиц CELLPADDING. Он задает горизонтальный отступ между содержимым кадра и его границами. Наименьшее значение этого атрибута равно 1. Нельзя указать 0. Можно не присваивать ничего — по умолчанию атрибут равен 6.

Атрибут **marginheight**. Действует так же, как и **marginwidth**. Он задает поля в верхней и нижней частях фрейма.

Атрибут **scrolling**. Дает возможность пользоваться прокруткой во фрейме. Возможные варианты: **scrolling=yes**, **scrolling=no**, **scrolling=auto**. **Scrolling=yes** означает, что во фрейме всегда будут полосы прокрутки, даже если это не нужно. Если задать **scrolling=no**, полос прокрутки не будет, даже когда это необходимо. Если документ слишком большой, а вы задали режим без прокрутки, до-

кумент просто будет обрезан. Атрибут **scrolling=auto** предоставляет браузеру самому решать, требуются полосы прокрутки или нет. Если атрибут **scrolling** отсутствует, результат будет таким же, как при использовании **scrolling=auto**.

Атрибут **noresize**. Как правило, пользователь может, перемещая границу фрейма мышкой, изменить его размер. Это удобно, но не всегда. Иногда требуется атрибут **noresize**. Помните: все границы фрейма, для которого вы задали **noresize**, становятся неподвижными – соответственно, может оказаться так, что размеры соседних фреймов тоже станут фиксированными. Пользуйтесь этим атрибутом с осторожностью.

Атрибут src. Применяется в теге **frame** при разработке фреймовой структуры для того, чтобы определить, какая страница появится в том или ином кадре. Если вы зададите атрибут **src** не для всех фреймов, у вас возникнут проблемы. Даже если страницы, отображаемые во фрейме, выбираются в соседнем фрейме, вы должны по крайней мере задать для каждого фрейма начальную страницу. Если вы не укажете начальную страницу и **url**, фрейм окажется пустым, а результаты могут быть самыми неожиданными.

Атрибут **target**. Чтобы разобраться с атрибутом **target**, необходимо вернуться к простому примеру с кадром оглавления. Когда пользователь щелкает мышкой на одной из ссылок в левом фрейме, соответствующая страница должна появиться в правом фрейме, а оглавление остается неизменным. Чтобы этого добиться, нужно определить целевой фрейм **target**, в котором будет отображаться страница для каждого пункта оглавления. Целевые фреймы задаются в ссылках левого фрейма. Вот зачем всем кадрам во фреймовой структуре были присвоены имена. Правый фрейм называется **main**, так что нужно в каждой ссылке добавить атрибут **target**="**main**", в результате чего соответствующая страница появится во фрейме **main**. Обратите внимание: каждая ссылка содержит атрибут **target**="**main**", который по щелчку мыши отображает страницу во фрейме **main**.

Атрибут **target** можно задавать для нескольких различных тегов. При использовании в теге <**base**> он направляет все ссылки в определенный целевой

фрейм, если в дальнейшем не предусмотрено другое. Можно задать атрибут **target** в теге <**area**> в активном изображении или в теге <**form**>. Фреймы полезны для организации форм. Пользователи будут видеть одновременно и форму, и результат своего выбора. Обычно при щелчке мышью кнопки **submit** форма исчезает, и появляется страница с результатами выбора. Сочетание форм и фреймов может оказаться удобным способом навигации.

ТЕМА 7. КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ CSS [2]

Общие положения

Каскадные таблицы стилей впервые были реализованы в Internet Explorer 3.0, но информация о них в то время была большей частью противоречивой. При реализации Internet Explorer 4.0 были приняты во внимание рекомендации REC-CSS1 Консорциума W3 относительно каскадных таблиц стилей, датированные 17 декабря 1996 года. К настоящему времени они пересмотрены и известны как рекомендации по каскадным таблицам стилей, уровень 1, документ REC-CSS1-19990111 от 11 января 1999 года. В мае 1998 года Консорциум издал рекомендации по каскадным таблицам стилей, уровень 2, документ REC-CSS2-19980512, часть из которых реализована в Internet Explorer 4.01. В соответствии с этими рекомендациями и будет вестись описание каскадных таблиц стилей.

Каскадные таблицы стилей, уровень 1, представляют собой простую технологию определения и присоединения стилей к документам HTML. Стиль, говоря житейским языком, – это все то, что определяет внешний вид документа HTML при его отображении в окне браузера: шрифты и цвета заголовков разных уровней, шрифт и разрядка основного текста, задаваемого в тэге абзаца <p>, и т. д. Стиль задается по определенным правилам, а таблица стилей – набор правил отображения, применяемых в документе, к которому присоединена соответствующая таблица стилей.

Таблица стилей – это шаблон, который управляет форматированием тэгов HTML в Web-документе. Концепция таблицы стилей напомнит ему концепцию стилевых файлов редактора Microsoft Word: изменить внешний вид документа

Word можно простым изменением присоединенных к нему стилей. Точно так же изменить внешний вид документа HTML можно простым изменением присоединенной к нему таблицы стилей.

Почему в название таблиц стилей включено определение "каскадные"? Рекомендации Консорциума W3 позволяют использовать несколько таблиц стилей для управления форматированием одного документа HTML, а браузер по определенным правилам выстраивает приоритетность применения этих таблиц. Они выстраиваются неким "каскадом", по которому и "прокатывается" документ.

Для разработки таблицы стилей достаточно немного ориентироваться в языке HTML и быть знакомым с базовой терминологией настольных издательских систем. Как уже отмечалось выше, таблица стилей представляет собой набор правил форматирования элементов HTML. Эти правила достаточно просты и легко запоминаемы. Например, если необходимо, чтобы в документе все заголовки первого уровня отображались синим цветом и шрифтом с кеглем (размером) 16 пунктов, то в таблице следует задать правило:

```
h1 {color: blue; font-size: 16pt}
```

Любое правило каскадных таблиц стилей состоит из двух частей: селектора и определения. Селектором может быть любой тэг HTML, для которого определение задает, каким образом необходимо его форматировать. Само определение, в свою очередь, также состоит из двух частей: свойства и его значения, разделенных знаком двоеточия (:). Назначение свойства очевидно из его названия. В приведенном правиле селектором является элемент H1, а определение, записанное в фигурных скобках, задает значения двух свойств заголовка первого уровня: цвет шрифта (свойство **color**) определен как синий (значение **blue**) и размер шрифта (свойство **font-size**) определен в 16 пунктов (значение I6pt). В одном правиле можно задать несколько определений, разделенных символом точка с запятой (;), как это демонстрируется в приведенном примере.

Созданная только что таблица стилей влияет на форматирование элемента определенного типа: заголовок первого уровня. Ее комбинация с другими таб-

лицами стилей определяет окончательное представление документа при его просмотре в окне браузера.

Синтаксис правил каскадных таблиц стилей не чувствителен к регистру. Селекторы, свойства и их значения можно задавать как строчными, так и прописными буквами, или в смешанном порядке. Однако каскадные таблицы стилей чувствительны к синтаксису задания правил и правильности названий свойств, значений и селекторов. Любая грамматическая ошибка приводит к тому, что правило пропускается анализатором браузера, и никакого предупреждающего сообщения не появляется.

Авторы страниц HTML должны писать свои таблицы стилей, только если они хотят придать документу вид, отличный от вида, предоставляемого умалчиваемой таблицей стилей браузера.

Встраивание таблиц стилей в документ

Чтобы таблица стилей могла воздействовать на внешнее представление документа, браузер должен знать о ее существовании. Для этого ее необходимо связать с HTML-документом.

Существует четыре способа связывания документа и таблицы стилей:

1. Связывание – позволяет использовать одну таблицу стилей для форматирования многих страниц HTML.

2. Внедрение – позволяет задавать все правила таблицы стилей непосредственно в самом документе.

3. Импортирование – позволяет встраивать в документ таблицу стилей, расположенную на сервере.

4. Встраивание в тэги документа – позволяет изменять форматирование конкретных элементов страницы.

Связывание позволяет хранить таблицу стилей в отдельном файле и присоединять ее к документам с помощью тэга <**link**>, задаваемого в разделе <**head**>:

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

Связываемый файл содержит набор правил каскадных таблиц стилей, определяющих форматирование документа, и должен иметь расширение CSS. Связывание позволяет разработчику применить одинаковый набор правил форматирования к группе HTML-документов, что приводит к единообразному отображению различных документов и придает некоторую системность серверу разработчика.

При внедрении таблицы стилей в документ правила, ее составляющие, задаются в стилевом блоке, ограниченном тэгами <**styletype**="**text**/**css**"> и </**style**>, который должен размешаться в разделе <**head**> документа:

```
<head>
    <title>Пример использования CSS</title>

<style type="text/css">

b { text-transform: uppercase; }
p { background-color: lightgrey; text-align:center; }

</style>
</head>
```

В приведенном выше примере встроенная таблица стилей определяет отображение всех абзацев в документе (элемент p) на сером фоне с центрированными строками. Полужирный текст, определяемый любым элементом b (тэг <b>) документа, будет отображаться прописными буквами, даже если в документе он задан строчными.

В тэге <**style**> можно импортировать внешнюю таблицу стилей с помощью свойства **@import** таблицы стилей:

```
@import: url(styles_gallery.css);
```

Его следует задавать в начале стилевого блока или связываемой таблицы стилей перед заданием остальных правил. Значением свойства **@import** является URL-адрес файла таблицы стилей.

Последний способ задания значений свойств таблицы стилей предназначен для оперативного форматирования определенного элемента документа и

называется внедрением. Каждый тэг HTML имеет параметр **style**, в котором можно задать значения его свойств в соответствии с синтаксисом каскадных таблиц стилей. Например, в следующем примере задается форматирование заголовка первого уровня, определяющее его отображение шрифтом красного цвета:

```
<h1 style="color: red">Заголовок первого уровня</h1>
```

Заголовок отображается шрифтом красного цвета. Если связанные, внедренные и импортируемые таблицы стилей влияют на форматирование всех элементов документа, для которых определены в таблицах правила, то встраивание определений стилей в конкретный тэг влияет на отображение только элемента, определяемого данным тэгом.

Рекомендуется избегать встраивания в тэги документа определений форматирования, так как подобная техника лишает разработчика преимуществ задания таблиц стилей в отдельном файле или в головной части документа, где в случае необходимости, их можно легко и быстро откорректировать. При форматировании отдельных тэгов придется просмотреть весь документ целиком, что требует достаточно большой и кропотливой работы.

Все способы встраивания таблиц стилей свободно сочетаются в одном документе. Например, можно разработать главную таблицу стилей для всех документов и связывать ее с каждым HTML-документом. Импортируемая или внедряемая таблица стилей будет уточнять форматирование элементов конкретного документа, а встраиваемые в тэг определения форматирования будут уточнять их отображение.

Группирование и наследование

Правила каскадных таблиц стилей состоят из селектора и определения. Для уменьшения размеров таблиц стилей можно группировать разные селекторы в виде списка элементов страницы HTML, разделенных запятыми, если для них задается одно правило. Например, следующие правила

```
h1 {font-family: Arial}
h2 {font-family: Arial}
h3 {font-family: Arial}
```

можно сгруппировать и задать в виде одного правила со списком селекторов

```
h1,h2,h3 {font-family: Arial}
```

Аналогично группируются определения, только в списке они разделяются точками с запятой ";". Следующие правила форматирования заголовка первого уровня

```
h1 {font-weight: bold}
h1 {font-size: 14pt}
h1 {font-family: Arial}
```

можно задать в виде одного правила, сгруппировав определения:

```
h1 {
    font-weight: bold;
    font-size: 14pt;
    font-family: Arial;
}
```

Некоторые свойства имеют собственный синтаксис группирования, связанный с заданием значений нескольких свойств в одном. Например, предыдущий пример при использовании свойства **font** запишется так:

```
h1 {font: bold 14pt Arial}
```

При задании таблицы стилей можно свободно комбинировать все три правила группирования для уменьшения ее размеров.

Как будет отображаться элемент, расположенный внутри другого элемента страницы, если для последнего задано правило форматирования, а для вложенного элемента нет? Например, пусть цвет шрифта абзаца определен как синий: **p{color: blue}**. Как будет отображаться выделенный элемент текста, задаваемый тэгом <**em**>, если для него не определено правило форматирования? В подобных случаях вложенный элемент наследует правила форматирования элемента-родителя. В нашем примере выделенный элемент будет также ото-

бражаться синим цветом. Другие свойства ведут себя аналогично свойству color, например **font-family**, **font-size**.

Некоторые свойства не наследуются вложенными элементами от своих родителей, например свойство **background**, но по умолчанию вложенные элементы будут отображаться с фоном родительского элемента.

Наследование полезно при задании значений свойств, применяемых к документу по умолчанию. Для этого достаточно задать все свойства для элемента, порождающего все остальные элементы страницы HTML. Таким элементом является тело документа, определяемое тэгом <**body**> :

```
body {
    color: black;
    font-family: "Times New Roman";
    background: url(texture.gif) white;
}
```

Приведенные правила задают форматирование документа по умолчанию: черным шрифтом гарнитуры Times New Roman с фоном, задаваемым графическим файлом texture.gif, или на белом фоне, если файл не доступен.

Селекторы

Правила каскадных таблиц стилей, в которых в качестве селектора используются тэги HTML, влияют на отображение всех элементов заданного типа в документе. Следующее правило отображает без подчеркивания все ссылки (тэг <a>) в документе:

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

Если необходимо некоторые ссылки отобразить по-другому, то для них можно применить параметр **class**, добавленный в HTML 4.0 в качестве стандарта для всех тэгов. Значением параметра **class** является ссылка на класс, задаваемый в таблице стилей.

Класс позволяет задать разные правила форматирования для одного элемента определенного типа или всех элементов документа. Имя класса указывается в селекторе правила после имени тэга и отделяется от него точкой.

Можно определить несколько правил форматирования для одного элемента и с помощью параметра **class** соответствующего тэга применять разные правила форматирования в документе. Например, можно определить два класса отображения заголовка первого уровня:

```
h1.red {color: red;}
h1.blueBgrd{color: red; background-color: blue}
```

В тексте документа ссылка на соответствующий класс задается в параметре **class**:

```
<h1 class="red">Красный шрифт</h1>
<h1 class="blueBgrd">Красный шрифт на синем фоне</h1>
```

Имя класса в параметре **class** задается без лидирующей точки. Оно может быть заключено в двойные или одинарные кавычки или задаваться вообще без кавычек, например **class=red**.

В приведенном примере классы задавались для разного отображения элементов одного типа. Если класс должен применяться ко всем элементам документа, то в селекторе задается имя класса с лидирующей точкой без указания конкретного элемента:

```
.red {color: red;}
.blueBgrd{color: red; background-color: blue}
```

Теперь два класса **red** и **blueBgrd** можно применять к любым элементам документа:

```
<p class="red">Первый абзац</p>
<p class="blueBgrd">Второй абзац</p>
```

Первый абзац отобразится красным шрифтом, а второй – красным шрифтом на синем фоне.

Параметр **id**, как и параметр **class**, не влияет на отображение браузером элемента HTML, но он задает уникальное имя элемента, которое используется для ссылок на него в сценариях и таблицах стилей. Параметр **id** можно применять к любому элементу документа, но только один раз.

Правила таблиц стилей регламентируют использование уникального идентификационного имени элемента в качестве селектора, предваряя его символом #:

```
<style type="text/css">
#par24 { letter-spacing: 1em; }
#form2{ color: red; background-color: blue}
</style>
</head>
<body>
<p id=par24>Разреженные слова в абзаце</p>
<h1 id=form2>черный шрифт</h1>
</body>
```

В этом примере абзац идентифицирован именем **par24** в параметре **id**, поэтому к нему применимо правило с селектором **#par24**.

Для организации связи с правилами форматирования следует избегать использования идентификаторов конкретных элементов страницы, задаваемых в параметре **id**. Если действительно необходимо отформатировать конкретный элемент, лучше использовать технику встраивания правил форматирования в тэг.

При разработке страниц HTML часто приходится одни элементы вкладывать в другие, например, выделять слова тэгом <**em**> в каком-нибудь абзаце, задаваемом тэгом <**p**>.

```
<p>Идейные соображения высшего порядка, а также
консультация с широким активом представляет собой
интересный <em>эксперимент</em> проверки форм развития.
Товарищи! консультация с широким активом представляет
собой интересный <em>эксперимент</em>.</p>
```

В этом случае говорят, что элемент **p** порождает элемент **em** и является его предком, а сам элемент **em** является потомком элемента **p**. Некоторые свойства предка наследуются потомком, например, цвет шрифта (свойство **color**). Чтобы вложенные элементы отображались со своими значениями свойств, можно определить для них правила форматирования, как показано ниже:

```
p em {color: yellow}
```

Контекстный селектор состоит из нескольких простых, разделенных пробелами. Интерпретатор браузера просматривает в стеке все открытые элементы,

находит элементы **ем**, порожденные элементом **р**, и применяет к ним указанное правило форматирования.

Таким образом, правила с контекстными селекторами задают исключения из общих правил форматирования элементов документа, определенных с простыми селекторами.

Псевдоклассы

Обычно правила форматирования присоединяются к элементу страницы, имеющему определенное положение в структуре документа. Концепция псевдоклассов и псевдоэлементов расширяет адресацию правил форматирования, позволяя внешней информации влиять на процесс форматирования. Псевдоклассы и псевдоэлементы можно использовать в селекторах правил форматирования, однако реально в исходном тексте документа HTML они не существуют. Вернее, они как бы вставляются браузером при определенных условиях в документ, и на них можно ссылаться в таблицах стилей. Их называют классами и элементами, так как это удобный способ описания их поведения. Говоря точнее, их поведение определяется фиктивной последовательностью тэгов.

Псевдоэлементы применяются для адресации некоторых частей элементов, тогда как псевдоклассы позволяют таблицам стилей для элементов разных типов применять разные процедуры форматирования.

Связь в документе HTML определяется тэгом <**а**> с параметром **href**. Обычно браузеры отображают посещенные связи отлично от непосещенных (например, разными цветами). Уровень 1 каскадных таблиц стилей регламентирует правила для отображения связей через псевдоклассы элемента **а**:

```
a:link { color: red } /* непосещенная связь */
a:visited { color: blue } /* посещенная связь */
a:active { color: green } /* активная связь */
a:hover { color: lime } /* связь, на которой расположен
курсор мыши */
```

Любую связь в документе можно отнести к одному из перечисленных классов. Активная связь – это связь, которая выбрана в данный момент.

Комментарии в каскадных таблицах стилей задаются с помощью символов /* и */.

Свойства форматирования элементов

В каскадных таблицах стилей все доступные свойства форматирования элементов в документе HTML разбиты на 9 категорий, представленных в таблице10.

Таблица 10. Категории свойства элементов

Категория	Устанавливает
Шрифт	Типографские свойства шрифтов
Цвет и фон	Цвет текста и фона, а также картинки в качестве фона
Текст	Выравнивание, форматирование и разрядку текста
Блок	Свойства форматирования блоковых элементов
Визуальное форматирование	Свойства, связанные с блоками отображения элементов, их позиционированием и отображением списков
Печать	Спецификацию разрыва страницы
Фильтры и переходы	Мультимедийные эффекты и преобразования графических изображений
Псевдоклассы и другие свойства	Свойства @import, cursor и important

Прежде чем переходить к описанию свойств форматирования, остановимся на единицах измерения, используемых для задания значений свойств.

Для задания значений свойств, определяющих некоторые размеры, в каскадных таблицах стилей применяются относительные и абсолютные единицы измерения длины. Относительные единицы задают длину относительно значения другого свойства, определяющего длину. Документы, в которых таблицы стилей используют относительные единицы измерения, более приспособлены для отображения на разных устройствах. Абсолютные единицы измерения полезны только тогда, когда известны физические характеристики устройства отображения.

В таблице11 отображены единицы измерения, используемые в таблицах стилей.

Таблица 11. Единицы измерения в таблицах стилей

Единицы измерения	
Относительные	**Абсолютные**
em Высота шрифта элемента ex Высота буквы px Пиксел % Процент	in Дюйм (1 in = 2.54 cm) cm Сантиметр mm Миллиметр pt Пункт (1pt= 1/72 in) pc Пика (1 pc = 12 pt)

Относительные единицы измерения em и ex во всех свойствах вычисляются относительно высоты шрифта элемента. Единственное исключение – свойство **font-size**, в котором эти единицы относятся к высоте шрифта элемента-родителя.

Пиксели являются единицами измерения, относящимися к разрешению дисплея компьютера. Если плотность пикселей на устройстве вывода сильно отличается от типового дисплея, браузер должен переопределить эту единицу.

В качестве значений цветов можно использовать зарезервированные ключевые слова HTML для определения наиболее употребительных цветовых оттенков (например, aqua, black, white и т. п.) или использовать цветовую модель RGB. Примеры способов задания цветовых оттенков приведены ниже:

```
em { color: #f00 } /* #rgb соответствует ttrrggbb */
em { color: #ff000000 } /* #rrggbb */
em { color: rgb(255,0,0) } /* целые в интервале 0 - 255 */
em { color: rgb(100%, 0%, 0%) } /* вещественные от 0.0% до
                                   100.0% */
```

Форматирование шрифта

Выбор подходящего шрифта для отдельных частей документа является одним из наиболее часто выполняемых действий в процессе разработки HTML-документа. Шрифты различаются по своему внешнему виду (начертанию), размеру, стилю (прямой, курсив или наклонный) и по "жирности" отображения (нормальный, полужирный). Каскадные таблицы стилей предоставляют в рас-

поряжение разработчика набор свойств для установки всех перечисленных параметров шрифтов.

Свойство **font-family** задает приоритетный список семейств шрифтов и/или типовых семейств шрифтов. Если использовать для отображения страницы один определенный шрифт, то может оказаться, что этот шрифт не поддерживает некоторые символы, содержащиеся на странице, или на компьютере пользователя нет вообще этого шрифта. Для разрешения подобных проблем это свойство позволяет разработчику страницы задать список шрифтов одного стиля и размера, среди которых браузер может искать необходимый символ. В отличие от других свойств каскадных таблиц стилей названия семейств в списке отделяются запятыми, чтобы показать их альтернативность:

```
body{font-family: TimesDL, "Times New", serif}
```

При интерпретации HTML-страницы браузер сначала ищет на компьютере пользователя шрифт TimesDL. Если такой шрифт отсутствует, то браузер пытается применить шрифт Times New, а если и он не найден, то используется любой шрифт из семейства шрифтов serif – одного из типовых семейств шрифтов компьютера.

Понятие типовых семейств шрифтов введено в каскадные таблицы стилей с целью реализации наихудшего варианта отображения страницы, если не найдены специально использованные автором шрифты. В любой реализации каскадных таблиц стилей должны существовать пять типовых семейств шрифтов, которые соответствуют реальным шрифтам, обычно устанавливаемым на большинстве компьютеров: serif (например, Times), sans-serif (например, Helvetica), cursive (например, Zapf-Chancery), fantasy (например, Western), monospace (например, Courier).

Имена шрифтов, состоящих из нескольких слов, должны заключаться в кавычки:

```
body {font-family: "Times New Roman", serif}
<body style="font-family: 'Times New Roman', serif">
```

Следует использовать кавычки разных типов при задании последовательности всех определяемых свойств в параметре **style** и при задании имени шрифта в свойстве **font-family**.

Свойство **font-style** определяет стиль шрифта из выбранного семейства: нормальный (normal), курсивный (italic) или наклонный (oblique).

Нормальный шрифт – это обычный прямой шрифт, используемый для печати документов. Курсивный стиль шрифта напоминает каллиграфические этюды в прописях первоклассников и близок к рукописному. Наклонные шрифты генерируются из обычных прямых шрифтов небольшим наклоном символов.

Обычно в базе шрифтов браузера все шрифты, в именах которых встречаются слова Oblique, Slanted или Incline отмечены как наклонные (oblique) шрифты. Шрифты, в названиях которых присутствуют слова Italic, Cursive или Kursiv, отождествляются браузером с курсивными (italic).

Следующие правила определяют курсивный стиль шрифта заголовка первого уровня и нормальный, прямой шрифт выделенных частей заголовка:

```
h1 {font-style: italic}
h1 em {font-style: normal}
```

Каскадные таблицы стилей реализуют еще одну вариацию шрифта выбранного семейства – капитель (small-caps). В шрифте этого стиля все строчные буквы выглядят как прописные, но меньшего размера и с несколько измененными пропорциями.

Значение normal свойства **font-variant** не изменяет вида шрифта, а значение small-caps выбирает вариант капитель шрифта. Рекомендации по каскадным таблицам стилей допускают создание шрифта капитель простой заменой строчных букв масштабированными символами верхнего регистра.

Следующие правила задают отображение заголовка четвертого уровня капителью с наклонной капителью в выделенных частях:

```
h4 {font-variant: small-caps}
em {font-style: oblique}
```

Свойство **font-variant** наследуется элементом em от своего родителя, поэтому в приведенном примере выделенные части заголовка будут отображаться наклонной капителью.

Свойство **font-weight** выбирает из заданного семейства шрифт определенной жирности. В рекомендациях регламентируется 9 градаций жирности шрифта, задаваемых числами 100, 200 и так далее до 900. Значение 100 соответствует самому "бледному" шрифту, тогда как 900 – самому "жирному".

Для задания нормального шрифта используется ключевое слово normal, что соответствует цифровому значению 400. Значение bold применяется для выбора общепринятого полужирного начертания шрифта и его цифровым эквивалентом является 700.

Выбор определенной градации жирности шрифта не означает, что в семействе существует шрифт с заданной жирностью. Единственное, что гарантируется, – это то, что шрифт с большим значением жирности не светлее предыдущего значения. Некоторые семейства имеют только две градации жирности: нормальную и полужирную.

Свойство **font-size** определяет размер шрифта. Его значение может быть абсолютным или относительным.

Абсолютное значение можно задать одним из следующих ключевых слов: xx-small, x-small, small, medium, large, x-large, xx-large, которые ЯВЛЯЮТСЯ индексами в таблице размеров шрифтов, поддерживаемой браузером. По умолчанию браузер использует значение medium.

Абсолютное значение можно задать и в виде абсолютного значения длины, например 10pt, но в этом случае высота шрифта не зависит от хранимой таблицы размеров шрифтов браузера.

Относительный размер шрифта можно задать также в процентах к размеру шрифта родителя или в относительных единицах длины:

```
<style type="text/css">

    p {font-size: 10pt}
    em {font-size: 120%}
    em {font-size: 1.2em}

</style>
</head>
<body>

<p> В абзац можно помещать <em>несколько</em>
    элементов.</p>
```

Два последних правила для выделенного в абзаце элемента **em** определяют одинаковую высоту шрифта 12pt.

Основное назначение свойства **font** – установить в одном определении значения свойств font-style, font-variant, font-weight, font-size, line-height и font-family. Все значения перечисленных свойств задаются через пробелы в том порядке, как они перечислены выше. Первые три свойства могут не задаваться, что соответствует установке их значений в normal. Размер шрифта и высота строки (свойство line-height) задаются через косую черту. Элементы списка семейств шрифтов свойства font-family задаются через запятую:

```
p {font: oblique 12pt/14pt "Times Cyr", serif}
```

В этом примере для абзаца задается наклонный шрифт Times Cyr высотой 12 пунктов. Высота строк – 14 пунктов. Если не найден шрифт Times Cyr, то применяется любой шрифт типового семейства serif.

Для установки цвета текста элемента существует единственное свойство **color**. Его значением является цвет, задаваемый с помощью ключевых слов или RGB функции:

```
p {color: blue}
em{color: rgb(0,0,255)}
```

Оба правила в примере устанавливают синий цвет текста соответствующих элементов.

Форматирование фона

Для установки параметров фона элемента существует несколько свойств, задающих значения индивидуальных параметров фона, и свойство background, в котором можно установить одновременно все значения параметров фона.

Цвет фона определяется значением свойства **background-color**, а изображение, используемое в качестве фона, задается свойством **background- image**.

Начальным значением свойства **background-color** является transparent, которое определяет фон элемента как прозрачный. Значением свойства **background-image** является абсолютный или относительный адрес файла изображения, используемого в качестве фона. Если это свойство определено, то рекомендуется задать также и цвет фона, который будет использоваться в случае недоступности файла изображения.

```
body {
    background-color: lightsteelblue;
    background-image: url(/image/image.gif);
}
p{background-image: none;}
```

В приведенном примере задается адрес файла изображения для фона тела документа и явно указывается отсутствие фона для абзацев документа.

Если фон задан в виде изображения, то свойство **background-repeat** определяет его повторяемость и способы повторяемости. Допустимыми значениями являются repeat (повторяемость и по вертикали, и по горизонтали), repeat-x и repeat-y (повторяемость соответственно по горизонтали или вертикали) и no-repeat (изображение не повторяется). В следующем примере задается повторяемость изображения фона по вертикали.

```
body {
    background-color: lightsteelblue;
    background-image: url(/image/image.gif);
    background-repeat: repeat-y;
}
```

Свойство **background-attachment** определяет, будет ли фон, на котором отображается документ, оставаться неподвижным при прокрутке содержимого

окна браузера или он будет прокручиваться вместе с документом. В первом случае реализуется эффект перемещения содержимого окна над неподвижным рисунком. Значение fixed оставляет фон неподвижным, а значение scroll заставляет его перемещаться вместе с содержимым документа при прокрутке. Пример закрепленного в окне браузера изображения фона представлен ниже:

```
body {
    background-color: lightsteelblue;
    background-image: url(/image/image.gif);
    background-repeat: repeat-y;
    background-attachment: fixed;
}
```

Свойство **background-position** определяет начальное положение изображения, используемого в качестве фона, в блоке содержимого элемента. Значением этого свойства являются координаты привязки определенных точек изображения и блока содержимого. Их можно задавать в процентах, в абсолютных единицах длины, а также с использованием комбинаций ключевых значений.

Пара 0%0% означает, что верхний левый угол изображения помещается в верхний левый угол блока содержимого элемента (это значение является значением по умолчанию). Пара 100% 100% размещает нижний правый угол изображения в нижний правый угол блока содержимого. Пара значений, отличных от указанных, например 10% 80%, помещает точку изображения, расположенную на расстоянии в 10% ширины от левого края и в 80% высоты от верхнего края, в точно такую же точку блока содержимого элемента.

Пара абсолютных значений, например 10mm 10mm, размещает верхний левый угол изображения на 10 мм правее и на 10 мм ниже левого верхнего угла блока содержимого.

Ключевые значения и их допустимые комбинации вместе с эквивалентными числовыми значениями представлены в таблице 12.

Таблица 12. Допустимые комбинации ключевых значений

Комбинация	Значение
top left, left top	0% 0%
top, top center, center top	50% 0%
top right, right top	100% 0%
left, center left, left center	0% 50%
center, center center	50% 50%
right, center right, right center	100% 50%
bottom left, left bottom	0% 100%
bottom, bottom right, bottom center	50% 100%
bottom right, right bottom	100% 100%

Свойство **background** позволяет одновременно устанавливать значения свойств background-color, background-image, background-repeat и background-attachment. Все допустимые значения индивидуальных свойств задаются в виде списка, элементы которого отделены пробелами. Если значение какого-либо свойства не задано, то оно устанавливается в начальное значение, определяемое браузером:

```
body {
    background: lightsteelblue url(/image/image.gif) center
}
```

Это правило устанавливает цвет и изображение фона, а также положение изображения в окне браузера. Остальные свойства фона принимают начальные значения.

Форматирование текста

Свойства данной категории влияют на отображение символов, слов и абзацев. Они определяют расстояние между словами и буквами в словах, задают отступы и высоту строк в абзацах.

Свойство **letter-spacing** влияет на расстояние между символами при отображении текста. Его значение, задаваемое в единицах длины, определяет пробел, добавляемый к установленному по умолчанию пробелу между символами.

На рис. 12 показано отображение текста с установками по умолчанию и с увеличенным на 0.5em пробелом между символами:

```
p{letter-spacing: 0.5em}
```

Слово Слово Слово

С л о в о С л о в о С л о в о

Рис. 12. Абзац с увеличенным расстоянием между символами

Браузер увеличивает не только расстояние между символами слов, но и расстояние между словами.

Каскадные таблицы стилей позволяют преобразовывать текст. Если значение свойства **text-transform** равно capitalize, то все слова отображаются с прописной буквы. Значения uppercase и lowercase этого свойства приводят, соответственно, к преобразованию всех букв в прописные или строчные, независимо от их задания в тексте документа HTML.

Свойство **text-decoration** задает подчеркивание, подчеркивание или перечеркивание текста. Соответствующие значения этого свойства следующие: underline, overline и line-through.

Выравнивание текста в блоке содержимого элемента определяется значением свойства **text-align**. Текст выравнивается относительно блока содержимого элемента по левому краю при значении left, по правому краю – при значении right и по центру – при значении center.

Отступ первой строки элемента задается значением свойства **text-indent**, которое определяет величину отступа в абсолютных или относительных единицах длины.

Свойство **vertical-align** определяет положение элемента по вертикали относительно элемента-родителя. Его значением может быть любое ключевое слово из таблицы 13.

Таблица 13. Ключевые значения выравнивания по вертикали

Значение	Результат
baseline	Выравнивание базовой линии (или низа, если элемент не имеет базовой линии) по линии родителя
middle	Выравнивание средней точки элемента (обычно изображения) на уровне базовой линии родителя плюс половина ширины блока содержимого родителя
sub	Элемент отображается как нижний индекс
super	Элемент отображается как верхний индекс
text-top	Выравнивание верха элемента с верхом шрифта элемента-родителя
text-bottom	Выравнивание верха элемента с низом шрифта элемента-родителя
top	Выравнивание верха элемента с верхом самого высокого элемента строки
bottom	Выравнивание низа элемента с низом всех расположенных элементов строки

Значения этого свойства, заданные в виде процентов, вычисляются относительно высоты строки (свойство **line-height**) самого элемента. Они поднимают базовую линию (или низ элемента, если он не имеет базовой линии) на заданную высоту относительно базовой линии элемента-родителя, если значение положительно, и опускают, если значение отрицательно.

Расстояние между базовыми линиями двух соседних строк (высота строки) задается установкой значения свойства **line-height**. Числовое значение этого свойства определяет высоту строки, вычисляемую умножением размера шрифта текущего элемента на заданное число.

Все текстовые свойства, кроме СВОЙСТВ **text-decoration** и **vertical-align**, наследуются элементами-потомками от родителей.

ТЕМА 8. БЛОЧНОЕ ПРЕДСТАВЛЕНИЕ ЭЛЕМЕНТОВ НА СТРАНИЦЕ

Ранее был рассмотрен табличный способ верстки страниц. В этом способе для задания структуры сайта используется тег <**table**> и его дочерние теги. Верстка с помощью таблиц позволяет наиболее пропорционально расположить

все элементы дизайна относительно друг друга. Однако табличная верстка обладает рядом недостатков:

- Содержимое страницы, сверстанной на основе таблиц, не будет отображено до тех пор, пока не загрузятся все данные.
- Плохая индексация, объясняется большими промежутками между блоками текста, расположенного в разных ячейках таблицы.

Теперь табличная верста редко используется в качестве основного метода создания сайтов. Сейчас ее применяют лишь для структурирования табличных данных и расположения графических изображений.

Более популярен блочный способ верстки Web-страниц. В отличие от табличной блочная верстка обладает рядом преимуществ[30]:

- отделение стиля элементов от кода html;
- возможность наложения одного слоя на другой – такая возможность во многом облегчает позиционирование элементов;
- лучшая индексация поисковиками;
- Высокая скорость загрузки страницы, состоящей от взаимно независимых элементов;
- легкость создания визуальных эффектов (выпадающих меню, списков, всплывающих подсказок).

Основным недостатком блочной верстки является разная интерпретация кода различными браузерами. Поэтому часто html страницы приходится изменять и использовать специальные приемы.

С появлением блочной верстки родилось такое понятие, как «кроссбраузерность». Из-за различия отображения одного и того же элемента в разных браузерах верстальщикам приходится вставлять в основной html целые дополнительные фрагменты кода для разных браузеров.

Основным элементом, применяемым в блочной верстке, является тег <**div**>. Участок кода, отделенный этим тегом, называется блоком. Все стилевые решения вынесены за границы кода html в каскадные таблицы стилей.

Перед началом верстки готовый **psd** макет сайта в графическом редакторе разрезают на блоки (слои). В отдельную папку помещают вырезанные фоновые картинки, которые будут прикрепляться отдельно к каждому слою.

Для определения блочной модели в форматировании имеется свойство **display**, значение которого определяет, отображается или не отображается (**none**) элемент, является ли он блоком (**block**), списком (**list-item**) или встроенным элементом (**inline**).

Элементы со значением свойства **display** равным **block** или **list-item**, а также элементы со значением свойства **float**, отличным от **none** (не "плавающие" элементы), являются блоковыми элементами. Их форматирование связано с установкой значений соответствующих параметров вложенных блоков, составляющих элемент в целом. На рис. 11 показаны все параметры, доступные в модели форматирования каскадных таблиц стилей для блоковых элементов.

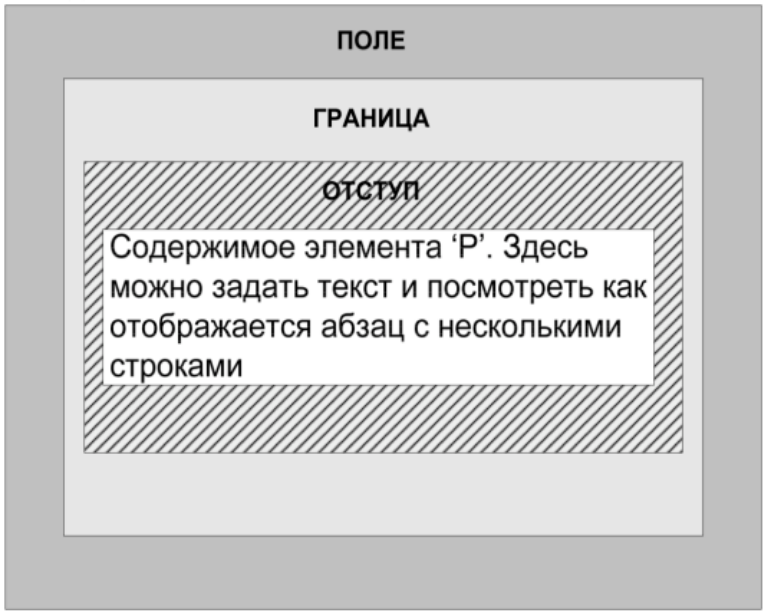


Рис. 11. Блоковая модель форматирования абзаца

Среди блоковых элементов можно выделить три большие группы, форматирующие блоки полей, границ и отступы, причем группу, работающую с границей, можно подразделить еще на четыре группы, устанавливающие значения цвета, стиля, ширины и одновременно всех перечисленных свойств границы. Все эти группы отличает то, что в них включены свойства для задания парамет-

ров верхних, нижних, правых, левых и одновременно всех четырех частей соответствующих блоков форматирования элемента.

В группу форматирования поля входят свойства, устанавливающие ширину верхнего (**margin-top**), правого (**margin-right**), нижнего (**margin- bottom**) и левого (**margin-left**) поля элемента. В свойстве **margin** можно одновременно установить значения всех четырех параметров поля элемента.

Ширина соответствующих полей задается значением длины или в процентах от ширины ближайшего элемента-родителя. Начальные значения всех полей равны 0.

Если в свойстве **margin** заданы четыре значения, то они, соответственно, относятся к верхнему, правому, нижнему и левому полю. Если определено только одно значение, то оно применяется ко всем сторонам поля элемента.

При задании двух или трех значений недостающие значения берутся из установок противоположных сторон. Например:

```
body {margin: 1em 2em}
    /* верх и низ = 11em, право и лево = 2em */
```

Ширина верхнего, правого, нижнего и левого отступа определяется значением, соответственно, свойств **padding-top**, **padding-right**, **padding-bottom** и **padding-left**. Свойство **padding** позволяет одновременно установить значения всех четырех отступов элемента. Все, что было сказано о задании значений для одновременной установки полей, относится и к этому свойству.

Ширину верхней, правой, нижней или левой границы задают соответственно свойствами **border-top-width**, **border-right-width**, **border-bottom-width** и **border-left-width**. Значения свойства **border-width** определяют ширину границы элемента для всех перечисленных ее частей. Все, что было сказано о задании значений для одновременной установки полей, относится и к этому свойству.

Значениями этих свойств могут быть ключевые параметры thin, medium и thick или значение длины. Ширина границы, определяемая ключевыми параметрами, зависит от браузера. Единственное, что можно гарантировать, – это

то, что ширина thin не больше ширины medium, которая, в свою очередь, не больше ширины thick.

Цвета частей границы задаются значениями свойств **border-top-color**, **border-right-color**, **border-bottom-color**и**border-left-color**. Свойство **border-color** определяет цвета всех частей границы. Четыре параметра цвета подчиняются все тем же правилам, описанным при задании полей элемента.

Если задан тип границы (таблица 14), но не задан цвет границы, то по умолчанию используется цвет самого элемента.

Все предыдущие установки свойств границы не будут иметь никакого воздействия на отображение элемента, если не установлен тип границы, так как по умолчанию тип границы не определен, и она не отображается.

Для задания типа любой из четырех частей границы применяются свойства **border-top-style**, **border-right-style**, **border-bottom-style** и **border-left-style**.

Свойство **border-style** определяет одновременно типы всех частей границы. Значениями этих свойств могут быть ключевые параметры none, solid, double, groove, ridge, inset, outset. Типы Границ, соответствующих всем перечисленным значениям, представлены в таблице 14.

Таблица 14. Типы линий границы

Ключевой параметр	**Тип границы**
none	Граница не отображается (свойство border-width игнорируется)
solid	Граница отображается сплошной линией
double	Граница отображается двойной линией (сумма толщины двух линий и промежутка между ними равна значению атрибута border-width)
grove	Граница отображается, как будто она вдавлена в лист («желобок»)
ridge	Граница отображается, как будто она выдавлена из листа («барельеф»)
inset	Весь блок отображается, как будто он вдавлен в лист («желобок»)
outset	Весь блок отображается, как будто он выдавлен из листа («барельеф»)

Последняя большая группа свойств позволяет установить ширину, тип и цвет частей границы или всей границы в целом. Свойства **border-top**, **border-right**, **border-bottom** и **border-left** определяют ширину, тип и цвет, соответственно, верхней, правой, нижней и левой границы. Свойство **border** определяет одновременно параметры всех частей границы. В отличие от аналогичных свойств, задающих параметры полей и отступов, данное свойство устанавливает одинаковые значения для всех частей границы.

Визуальное форматирование блоков

Каскадные таблицы стилей предоставляют разработчику Web-документов мощные средства компоновки элементов HTML на странице документа, определяющие внешний вид страницы HTML в окне браузера.

Элементы HTML отображаются браузером последовательно в том порядке, как они определены в тексте HTML-файла с учетом их положения в структуре документа и расположения предыдущих отображенных элементов и элементов-контейнеров, в которых они могут содержаться. При компоновке страницы используются установки браузера для определения положения каждого элемента. Например, два последовательных абзаца следуют друг за другом, причем каждый начинается с новой строки.

Свойство **position** элемента позволяет определить способ его позиционирования на странице: статический, относительный или абсолютный. Относительный способ определяет смещение элемента относительно его естественного положения в потоке отображения элементов. Абсолютный способ удаляет элемент из естественного потока позиционирования и позволяет разместить его на странице абсолютно произвольным образом. Статический способ, являющийся умалчиваемым способом позиционирования элементов, предполагает естественный поток отображения элементов страницы в окне браузера в соответствии с иерархией объектов документа.

Значения static, relative и absolute свойства **position** определяют соответствующий способ позиционирования элемента, который, в конечном счете, скла-

дывается из значения указанного свойства элемента, его положения в иерархической структуре документа, местом его определения в исходном файле HTML и значениями его свойств top и left.

Абсолютно (absolute) позиционированный элемент и все его потомки изымаются из естественного потока отображения элементов и позиционируются независимо, причем сам элемент или его потомки могут перекрывать ранее отображенные элементы.

Чтобы определить точку отсчета местоположения элемента, следует найти его ближайшего родителя, позиционированного абсолютно или относительно. Положение левой верхней вершины блока этого элемента и будет точкой отсчета для абсолютно позиционированного элемента. Если процесс поиска подобного родителя (следует пропускать все позиционированные статически элементы) дойдет до элемента <**body**>, то тело документа и будет тем элементом, относительно которого позиционируется исходный элемент.

Еще два параметра, влияющие на отображение абсолютно позиционированного элемента, – это свойства **width** и **height** элемента. При абсолютном позиционировании элемент изымается из стандартного потока и отображается самостоятельно в своем собственном прямоугольном блоке, левый верхний угол которого помещается в определенную точку окна браузера. Свойства **width** и **height** задают ширину и высоту этого блока. Если они не заданы, то по горизонтали блок распространяется до правого края окна браузера, а по вертикали – на столько, на сколько необходимо для отображения содержимого элемента. Установка значений свойств **width** и **height** ограничивает размеры блока абсолютно позиционированного элемента. Если его содержимое не помещается в блок заданного размера, то оно просто невидимо для пользователя. Динамическим изменением размеров блока можно сделать так, что будет видно все содержимое элемента.

Относительно позиционированные и статически позиционированные элементы после изъятия из исходного текста документа всех абсолютно позиционированных элементов (вместе с их потомками) образуют непрерывный поток

отображения, в котором каждый последующий элемент позиционируется относительно конца предыдущего.

Относительно позиционированные элементы, являющиеся потомками абсолютно позиционированных элементов, также позиционируются в конец своего элемента-родителя.

```
p{
    position:relative;
    background-color:#90EE90;
}
```

Несколько свойств каскадных таблиц стилей позволяют организовать совместно со встроенными сценариями динамическое отображение и скрытие элементов страницы HTML.

Свойство **visibility** управляет отображением элемента. Если его значение равно visible (значение по умолчанию), то элемент отображается, если оно установлено равным hidden, то элемент не отображается. Когда для скрытия элемента используется его свойство **visibility**, то элемент не изымается из потока отображения. Это означает, что соответствующий ему блок занимает надлежащее положение на странице, но содержимое этого блока (элемент) не отображается.

Подобное поведение отличается от поведения объекта со свойством **display** равным none. В последнем случае элемент не только не отображается, но и изымается из потока отображения, и на странице нет пустого блока, соответствующего этому элементу.

Свойство **z-index** задает слой, в котором располагается элемент при отображении. Если слой расположен ближе к пользователю (значение свойства **z-index** больше), то элемент перекрывает любой другой элемент с меньшим значением слоя, даже если последний и отображается позже.

Если значением свойства **z-index** является auto, то элемент перекрывает все элементы с таким же значением этого свойства, но его перекрывает любой элемент со значением свойства **z-index**, отличным от auto.

Элементы, которые не форматируются как блоковые, являются встроенными (**inline**). Они совместно с другими элементами используют область строки. Обычно, выделяемые в строке элементы (**ем**, **strong**, и т. д.) классифицируются как встроенные.

Рассмотрим пример задания встроенных элементов в блоковый элемент абзац:

```
<p> В абзац можно помещать <b style= 'background-color :
lightsteelblue'> встраиваемые элементы</b> Их может быть
<em style='background-color: lightblue'>несколько</em>.</p>
```

Если ширина абзаца достаточна для отображения всех встроенных элементов в строке, то они отобразятся в одной строке. Если ширина недостаточна, то некоторые элементы могут быть разбиты на два и отображены в двух строках (рис. 12).

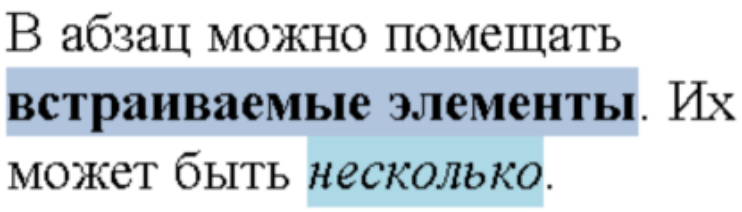


Рис. 12. Отображение встраиваемых элементов

Тема 9. Элементы анимации в HTML

Часто CSS применяют и для добавления движений на Web-страницы. Анимация может быть мощным инструментом, который позволяет посетителям сайта и клиентам легче взаимодействовать с интерфейсом продукта и быстрее добиваться своих целей. Этого можно достичь, если следовать в своей работе некоторым основополагающим принципам[35].

Принципы анимации[35]

Компания Disney занимается анимацией уже много лет и выработала 12 принципов анимации. В 1981 была опубликована первая версия книги «Иллю-

зия жизни: Анимация Disney»[26]. В ней описывают то, как использовать анимацию для создания у зрителей ощущения реального мира.

Те же самые принципы можно использовать для интерпретации анимации при создании Web-страниц.

Сплющивание и растяжение. В мире существует понятие «массы тела» – и когда объект движется, его масса остаётся неизменной. Когда мяч ударится о землю, он станет немного шире и сплющится, потому что во время физического контакта масса объекта перераспределяется. Этот эффект наиболее полезен при создании физических объектов (например, люди, часы или подпрыгивающие мячи).

Технику «сплющивания и расширения» можно использовать на Web-страницах для создания ощущения наличия у элемента физической массы. Даже небольшие изменения могут создать тонкий, но привлекающий взгляд эффект.

Ожидание. Движения не происходят внезапно. В обычной жизни любому действию предшествует некая подготовка, будь то замедляющийся перед падением со ступенек мяч или человек, сгибающий колени перед прыжком.

Этот эффект можно использовать для того, чтобы сделать движения объектов на Web-странице более реалистичными. Само ожидание может быть реализовано, к примеру, в качестве лёгкого покачивания, которое позволяет пользователям понять, что происходит и помогает им легче следить за перемещениями объекта по экрану.

Фокусировка. Этот эффект заключается в определении объекта в качестве фокуса сцены, в то время как остальные объекты освобождают место для главного действия. Для создания такого эффекта нужный объект соответствующим образом выдвигают на передний план или маскируют сопутствующие элементы, чтобы сфокусировать взгляд пользователя на том, что он должен увидеть.

В терминах Web-ресурса это означает использование полупрозрачного затемнения для определённого контента. Наложение более тёмного слоя

на существующую страницу и последующее размещение контента на переднем плане фокусирует на нём всё внимание пользователей.

Другой подход заключается в использовании движений. Когда на странице одновременно движутся несколько объектов, довольно трудно понять, на какой из них обратить внимание. Если в такой ситуации остановить все объекты, кроме одного, то все внимание автоматически будет направлено на него.

Движения «полный вперёд» и «пошаговое изменение». Прямое движение характеризуется полной отрисовкой всех кадров анимации. В свою очередь, пошаговое изменение характеризуется созданием последовательности ключевых кадров, интервалы между которыми заполняются позднее, обычно с помощью вспомогательных средств.

Второй подход используется в большинстве Web-анимации: переходы между главными кадрами реализуются браузером, который интерполирует разницу между каждым из них и дорисовывает столько промежуточных кадров, сколько потребуется для того, чтобы сделать итоговую анимацию плавной.

Следование и захлёстывание. Не всегда действия происходят одновременно. Когда движущийся автомобиль резко тормозит, то его передняя часть резко наклоняется вниз, а водитель внутри продолжает двигаться, пока не произойдёт окончательная остановка. Этот эффект достигается с помощью движений следования и захлёстывания – они нужны, чтобы просигнализировать о том, что объект остановился. К примеру, элемент, помещенный в список, может немного проехать нужное положение, а затем вернуться на правильную позицию.

Эффект захлёстывания создаётся с помощью придания объектам различного темпа движения. Этот подход используется при переходах в операционной системе IOS – некоторые элементы двигаются с разной скоростью, в результате перемещения становятся более реалистичными, чем они были бы в случае движения с одинаковой скоростью. Комбинирование движений даёт пользователю время на то, чтобы осознать происходящие изменения.

На Web-страницах это обычно достигается комбинированием плавных переходов или CSS-анимацией с разной скоростью проигрывания.

Плавное ускорение и замедление. Объекты редко переходят мгновенно от состояния покоя сразу к максимальной скорости. Обычно на то, чтобы разогнаться им требуется какое-то время – так же, как и для того, чтобы остановиться. Без постепенного ускорения и замедления движения выглядят неестественно.

В терминологии CSS ускорение и замедление известно как easing или «функция плавности», которая описывает изменение скорости анимации.

Дуги. Объекты редко двигаются по прямой, как правило, движение идёт по изогнутой дуге. Добиться такого движения по дуге с помощью CSS можно несколькими способами. Один заключается в совмещении нескольких сюжетов анимации. К примеру, анимация подпрыгивающего мяча может быть сделана с помощью движений объекта вверх и вниз, и второе движение совмещено с передвижением ещё и вправо. Второй способ – поворот элемента. С помощью изменения центра вращения и выноса его за пределы изначального объекта также можно сформировать дугу.

Второстепенное действие. Основное действие, происходящее на экране, можно подчеркнуть «второстепенной» анимацией. Это может быть движение рук человека при ходьбе или поднимающаяся с земли пыль при каждом ударе мяча.

В случае Web-сайтов вторичные элементы могут «разъезжаться», чтобы выделить то, на что пользователю действительно нужно обратить внимание (пример использования – перетягивание элемента для вставки в середину списка).

Время выполнения. С помощью манипуляции с длительностью анимации можно сделать так, чтобы крупные объекты казались тяжёлыми, а небольшие – лёгкими. На Web-страницах добиться этого можно простым изменением значений длительности анимации и временем движения объекта. С помощью изменения длительности анимации можно выделить её среди контента Web-страницы или во время взаимодействия с пользователем.

Преувеличение. Этот приём часто используется для привлечения внимания к определённым действиям и придании им большей драматичности в мультфильмах. Волк, который пытается кого-то укусить открывает рот гораздо шире, чем обычно – так гораздо страшнее и внушительнее.

На Web-странице объекты можно увеличивать и уменьшать, чтобы выделить их и привлечь внимание пользователя. Например, при заполнении формы активное поле имеет смысл сделать больше, а неактивные в данный момент, наоборот, уменьшить.

Объем. При работе с объемными объектами необходимо учитывать правила перспективы – люди живут в трёхмерном мире, поэтому если что-то выглядит не так, как они ожидают, это кажется неправильным.

В современных браузерах есть хорошая поддержка трёхмерных трансформаций. Это значит, что разработчик может создать анимацию, которая будет поворачивать объекты и перемещать их по экрану, а уже сам браузер создаст плавные переходы между состояниями.

Использование принципов анимации позволяет Web-разработчикам создавать более качественные и приятно выглядящие анимации. Если анимация учитывает физический вес объекта, выделяет изменения, в ней используются вторичные движения, а ее время исполнения тщательно рассчитано – то это значительно улучшит общее впечатление пользователей от страницы.

CSS-анимация [45]

Создание анимации на Web-страницах без использования JavaScript стало возможным благодаря появлению новой версии CSS3. Новый стандарт подразумевает поддержку современными браузерами. Для проверки совместимости был разработан тест Acid3, который проверяет бразуер на поддержку CSS3.

Анимацию CSS можно разделить на два типа: анимационные переходы элементов и создание ключевых кадров. В отличие от CSS-переходов, при создании ключевых кадров, автоматически воспроизводятся и повторяются эффек-

ты на протяжении заданного времени, а также есть возможность останавливать анимацию внутри цикла.

CSS-анимация может применяться практически для всех html-элементов, а также для псевдоэлементов **:before** и **:after**.

Псевдоэлемент **:after** используется для вывода желаемого текста после содержимого элемента, к которому он добавляется, а псевдоэлемент :before – до содержимого элемента.

Рассмотрим создание анимации по ключевым кадрам. Создание анимации начинается с установки ключевых кадров правила **@keyframes**. Кадры определяют, какие свойства на каком шаге будут анимированы. Каждый кадр может включать один или более блоков объявления из одного или более пар свойств и значений. Правило **@keyframes** содержит имя анимации элемента, которое связывает правило и блок объявления элемента.

```
@keyframes shadow {
    from {text-shadow: 0 0 3px black;}
    50% {text-shadow: 0 0 30px black;}
    to {text-shadow: 0 0 3px black;}
}
```

Ключевые кадры создаются с помощью ключевых слов **from** и **to** (эквивалентны значениям 0% и 100%) или с помощью процентных значений. Также можно комбинировать ключевые слова и проценты. Если кадры имеют одинаковые свойства и значения, их можно объединить в одно объявление:

```
@keyframes move {
    from, to {
        top: 0;
        left: 0;
    }
    25%, 75% {top: 100%;}
    50% {top: 50%;}
}
```

Если 0% или 100% кадры не указаны, то браузер пользователя создает их, используя вычисляемые (первоначально заданные) значения анимируемого свойства. Если у двух ключевых кадров будут одинаковые селекторы, то последующий отменит действие предыдущего.

После объявления правила **@keyframes**, можно ссылаться на него в свойстве **animation**:

```
h1 {
    font-size: 3.5em;
    color: darkmagenta;
    animation: shadow  2s infinite ease-in-out;
}
```

Не рекомендуется анимировать нечисловые значения (за редким исключением), так как результат в браузере может быть непредсказуемым. Также не следует создавать ключевые кадры для значений свойств, не имеющих средней точки, например, для значений свойства **color:pink** и **color:#ffffff**, **width:auto** и **width:100px** или **border-radius:0** и **border-radius:50%**.

Свойство **animation-name** задаёт имя анимации. Имя анимации создаётся в правиле **@keyframes**. Рекомендуется использовать название, отражающее суть анимации, при этом можно использовать одно или несколько слов, связанных между собой при помощи пробела или символа нижнего подчеркивания_. Свойство не наследуется.

В качестве значений свойства **animation-name** может быть «имя анимации», которое связывает правило **@keyframes** с селектором. Значение none – означает отсутствие анимации. Также используется, чтобы отменить анимацию элемента из группы элементов, для которых задана анимация. Значение initial устанавливает значение свойства в значение по умолчанию. inherit – наследует значение свойства от родительского элемента.

```
div {animation-name: mymove;}
```

Свойство **animation-duration** устанавливает продолжительность анимации, задаётся в секундах (s) или миллисекундах (ms), отрицательные значения не допустимы. Не наследуется. Если для элемента задано более одной анимации, то можно установить разное время для каждой, перечислив значения через запятую.

```
div {
    animation-name: mymove;
    animation-duration: 2s;
}
```

Свойство **animation-timing-function** определяет изменение скорости от начала до конца анимации с помощью временных функций. Задаётся при помощи ключевых слов или кривой Безье cubic-bezier(x1, y1, x2, y2). Не наследуется. Значения свойства приведены в таблице 15.

Таблица 15. Значения свойства animation-timing-function

ease	Функция по умолчанию, анимация начинается медленно, разгоняется быстро и замедляется в конце. Соответствуетcubic-bezier(0.25,0.1,0.25,1).
linear	Анимация происходит равномерно на протяжении всего времени, без колебаний в скорости. Соответствует cubic-bezier(0,0,1,1).
ease-in	Анимация начинается медленно, а затем плавно ускоряется в конце. Соответствует cubic-bezier(0.42,0,1,1).
ease-out	Анимация начинается быстро и плавно замедляется в конце. Соответствует cubic-bezier(0,0,0.58,1).
ease-in-out	Анимация медленно начинается и медленно заканчивается. Соответствует cubic-bezier(0.42,0,0.58,1).
cubic-bezier(x1, y1, x2, y2)	Позволяет вручную установить значения от 0 до 1. На этом сайте вы сможете построить любую траекторию скорости изменения анимации.
step-start	Задаёт пошаговую анимацию, разбивая анимацию на отрезки, изменения происходят в начале каждого шага. Эквивалентно steps(1, start).
step-end	Пошаговая анимация, изменения происходят в конце каждого шага. Эквивалентно steps(1, end).
steps(количество шагов,start|end)	Ступенчатая временная функция, которая принимает два параметра. Количество шагов задается целым положительным числом. Второй параметр необязательный, указывает момент, в котором начинается анимация. Со значением start анимация начинается в начале каждого шага, со значением end – в конце каждого шага с задержкой. Задержка вычисляется как результат деления времени анимации на количество шагов. Если второй параметр не указан, используется значение по умолчанию end.
initial	Устанавливает значение свойства в значение по умолчанию.

```
div {
    animation-name: mymove;
    animation-duration: 2s;
    animation-timing-function: linear;
}
```

С помощью пошаговой анимации можно создавать эффект печатающегося текста или индикатора загрузки (рис.13).

Рис. 13. Индикатор загрузки

Свойство **animation-delay** игнорирует анимацию заданное количество времени, что даёт возможность по отдельности запускать каждую анимацию. Отрицательная задержка начинает анимацию с определенного момента внутри её цикла, т.е. со времени, указанного в задержке. Это позволяет применять анимацию к нескольким элементам со сдвигом фазы, изменяя лишь время задержки.

Чтобы анимация началась с середины, нужно задать отрицательную задержку, равную половине времени, установленном в **animation-duration**. Не наследуется.

```
div {
    animation-name: mymove;
    animation-duration: 2s;
    animation-timing-function: linear;
    animation-delay: 2s;
}
```

Свойство **animation-iteration-count** позволяет запустить анимацию несколько раз. Значение 0 или любое отрицательное число удаляют анимацию из проигрывания. Значение infinite позволяет проигрывать анимацию бесконечно. Не наследуется.

```
div {
    animation-name: mymove;
    animation-duration: 2s;
    animation-timing-function: linear;
    animation-delay: 2s;
    animation-iteration-count: 3;
}
```

Свойство **animation-direction** задает направление повтора анимации. Если анимация повторяется только один раз, то это свойство не имеет смысла. Не наследуется. Значения свойства представлены в таблице 16.

Таблица 16. Значения свойства animation-direction

alternate	Анимация проигрывается с начала до конца, затем в обратном направлении.
alternate-reverse	Анимация проигрывается с конца до начала, затем в обратном направлении.
normal	Значение по умолчанию, анимация проигрывается в обычном направлении, с начала и до конца.
reverse	Анимация проигрывается с конца.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
div {
    animation-name: mymove;
    animation-duration: 2s;
    animation-timing-function: linear;
    animation-delay: 2s;
    animation-iteration-count: 3;
    animation-direction: alternate;
}
```

Все параметры воспроизведения анимации можно объединить в одном свойстве – **animation**, перечислив их через пробел:

```
div {
    animation: mymove 2s linear 2s 3 alternate;
}
```

Для воспроизведения анимации достаточно указать только два свойства – **animation-name** и **animation-duration**, остальные свойства примут значения по умолчанию. Порядок перечисления свойств не имеет значения, единственное, время выполнения анимации **animation-duration** обязательно должно стоять перед задержкой **animation-delay**.

Свойство **animation-play-state** управляет проигрыванием и остановкой анимации. Остановка анимации внутри цикла возможна через использование этого свойства в скрипте JavaScript. Также можно останавливать анимацию при наведении курсора мыши на объект – состояние **:hover**. Не наследуется. Значение свойства paused останавливает анимацию, а running – проигрывает анимацию.

```
div:hover {animation-play-state: paused;}
```

Свойство **animation-fill-mode** определяет порядок применения определенных в **@keyframes** стилей к объекту. Не наследуется. Значения свойства приведены в таблице 17.

Таблица 17. Значения свойства animation-fill-mode

none	Значение по умолчанию. Состояние элемента не меняется до или после воспроизведения анимации.
forwards	Воспроизводит анимацию до последнего кадра по окончанию последнего повтора и не отматывает ее к первоначальному состоянию.
backwards	Возвращает состояние элемента после загрузки страницы к первому кадру, даже если установлена задержка animation-delay, и оставляет его там, пока не начнется анимация.
both	Позволяет оставлять элемент в первом ключевом кадре до начала анимации (игнорируя положительное значение задержки) и задерживать на последнем кадре до конца последней анимации.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
div {animation-fill-mode: forwards;}
```

Для одного элемента можно задавать несколько анимаций, перечислив их названия через запятую:

```
div {animation: shadow 1s ease-in-out 0.5s alternate,
move 5s linear 2s;}
```

CSS-анимация придает динамичность и оживляет Web-страницы, улучшая взаимодействие с пользователем.

Тема 10. Построение Web-страниц с помощью фрэймворка Bootstrap

Bootstrap – это популярный фреймворк HTML, CSS и JS для разработки интернет проектов и Web-сайтов. В Bootstrap написаны готовые стили и скрипты, для применения которых достаточно всего лишь прописать необходимые стилевые классы и атрибуты HTML-элементам. С помощью фреймворка Bootstrap легко адаптировать любой Web-проект и хорошо отображать его на любых устройствах.

Фреймворки – это программные продукты, которые упрощают создание и поддержку технически сложных или нагруженных проектов. Фреймворк, как правило, содержит только базовые программные модули, а все специфичные для проекта компоненты реализуются разработчиком на их основе. Тем самым достигается не только высокая скорость разработки, но и большая производительность и надежность решений.

При работе с фрэймворком Bootstrap, первое с чего необходимо начать – это создание визуального макета. Есть специальная сетка для программы Adobe Photoshop, которая помогает при последующей верстке.

Шаблоны различаются по ширине, все зависит от того, для какого устройства создается макет.

Возьмем, например, макет размером 1170px он подходит для просмотра сайта на мониторе с разрешением не меньше 1170px (рис. 14).

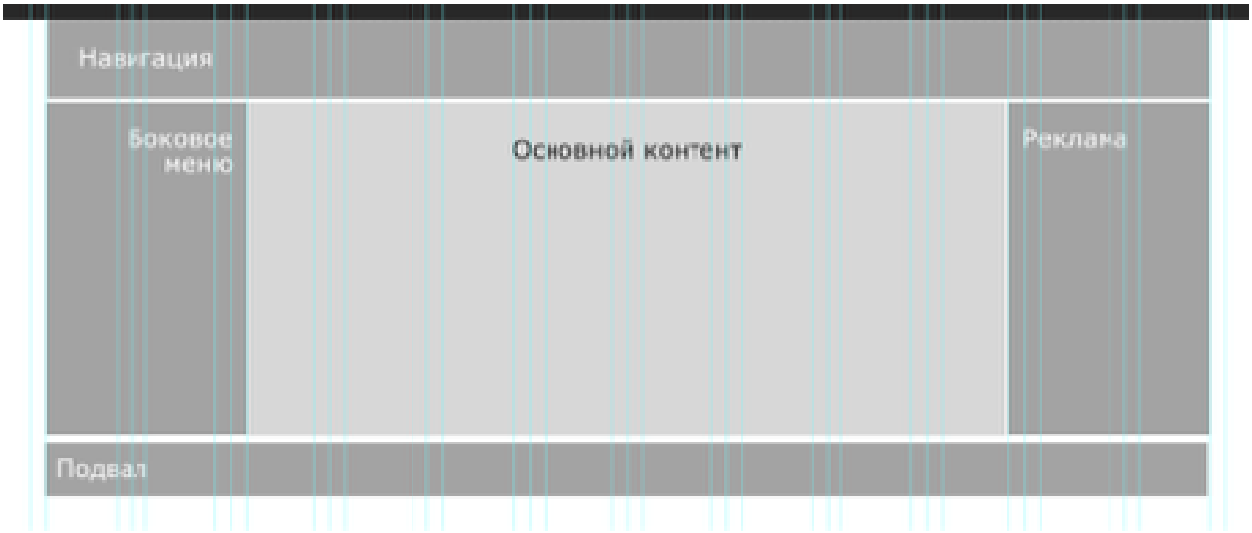


Рис. 14. Макет Web-страницы

Как видно из рис. 14, макет разделен на колонки (рис. 15), количество колонок всегда 12, ширина постоянна. Основная задача при создании макета размещать информацию и блоки в пределах заданных колонок.

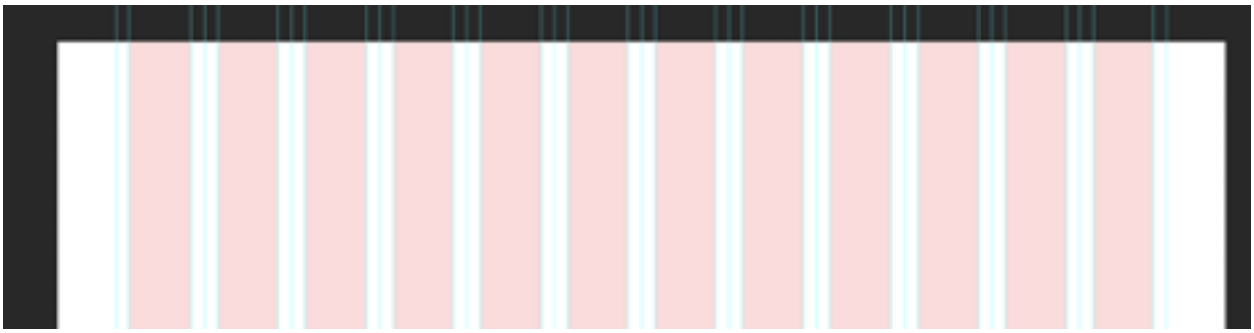


Рис. 15. Разбивка макета на 12 равных колонок

Из рисунка 14 видно, что навигация (меню) будет занимать все 12 колонок, в следующем ряду идут три колонки, левая и правая занимает по 2 колонки, а средняя 8. Подвал занимает, как и навигация, все 12 колонок.

Разделение на колонки очень важно, так как это будет определяющим при создании макета с использованием фреймворка Bootstrap.

Фреймворк Bootstrap загружается с официального сайта https://getbootstrap.com.В загруженных файлах будет структура и содержание в

соответствии с рис.16, сгруппированные логически по общим свойствам и содержащие обе версии: минимизированная и компилированная.

```
bootstrap/
+-- css/
|   +-- bootstrap.css
|   +-- bootstrap.min.css
+-- js/
|   +-- bootstrap.js
|   +-- bootstrap.min.js
+-- img/
|   +-- glyphicons-halflings.png
|   +-- glyphicons-halflings-white.png
L-- README.md
```

Рис. 16. Структура фреймворка Bootstrap

Для установки Bootstrap в заголовке Web-страницы необходимо подключить следующие файлы:

```
<link rel="stylesheet" href="css/bootstrap.min.css">
<script src="js/jquery.min.js"></script>
<script src="js/bootstrap.min.js"></script>
```

Контейнеры [36]

Контейнеры являются основными элементами макета в Bootstrap. Выбирайте отзывчый дизайн, фиксированную ширину контейнера (то есть max-width) или fluid-width (то есть 100% ширины области просмотра).

```
<div class="container">
  <!-- Content here -->
</div>
```

Используйте .**container-fluid** для того чтобы растянуть контейнер на всю ширину области просмотра.

```
<div class="container-fluid">
  ...
</div>
```

Система сетки Bootstrap [44]

Bootstrap включает отзывчивую, изначально ориентированную на мобильные устройства сетку, которая масштабируется, начиная с 12 столбцов как на устройствах или при изменении окна просмотра. Она включает предопределенные классы для простой настройки разметки, а также мощные примеси для генерации более семантической разметки .

Сетки используют для построения макетов страниц посредством компоновки строк и колонок, которые содержат контент.

Основы работы сетки Bootstrap:

- Row (строки) должны быть расположены внутри класса **.container**(fix) или **.container-fluid**(full-width) для правильного выравнивания и отступов.
- Используйте класс **.rows** (строки), чтобы создать горизонтальную группу колонок.
- Контент должен быть расположен внутри колонок, и только колонки могут быть первыми дочерними элементами rows (строки).
- Предопределенные классы сетки, например, такие как **.row** и **.col-xs-4**, позволяют быстро создать макет сетки.
- Колонки создают промежутки (зазоры между контентом колонок) посредством свойства CSS **padding**. Этот отступ смещается в строках для первой и последней колонки посредством отрицательного поля у элемента **.row**.
- Колонки сетки создаются при помощи указания одного номера из двенадцати возможных для колонок, которые вы хотите создать. Например, для построения трех равнозначных колонок достаточно использовать класс **.col-xs-4**.
- Если более чем 12 колонок расположены внутри одной строки, то каждая группа дополнительных колонок должна быть обернута новой строкой.
- Классы сетки подразделяются в зависимости от ширины устройств, которые определены в точках останова. При этом, например, применив класс **.col-md-*** к элементу его стиль будет использован не только для средних

устройств, но и к большим устройствам, но только если не задан класс **.col-lg-***.

На рис. 17 изображены принципы построения сетки Bootstrap.

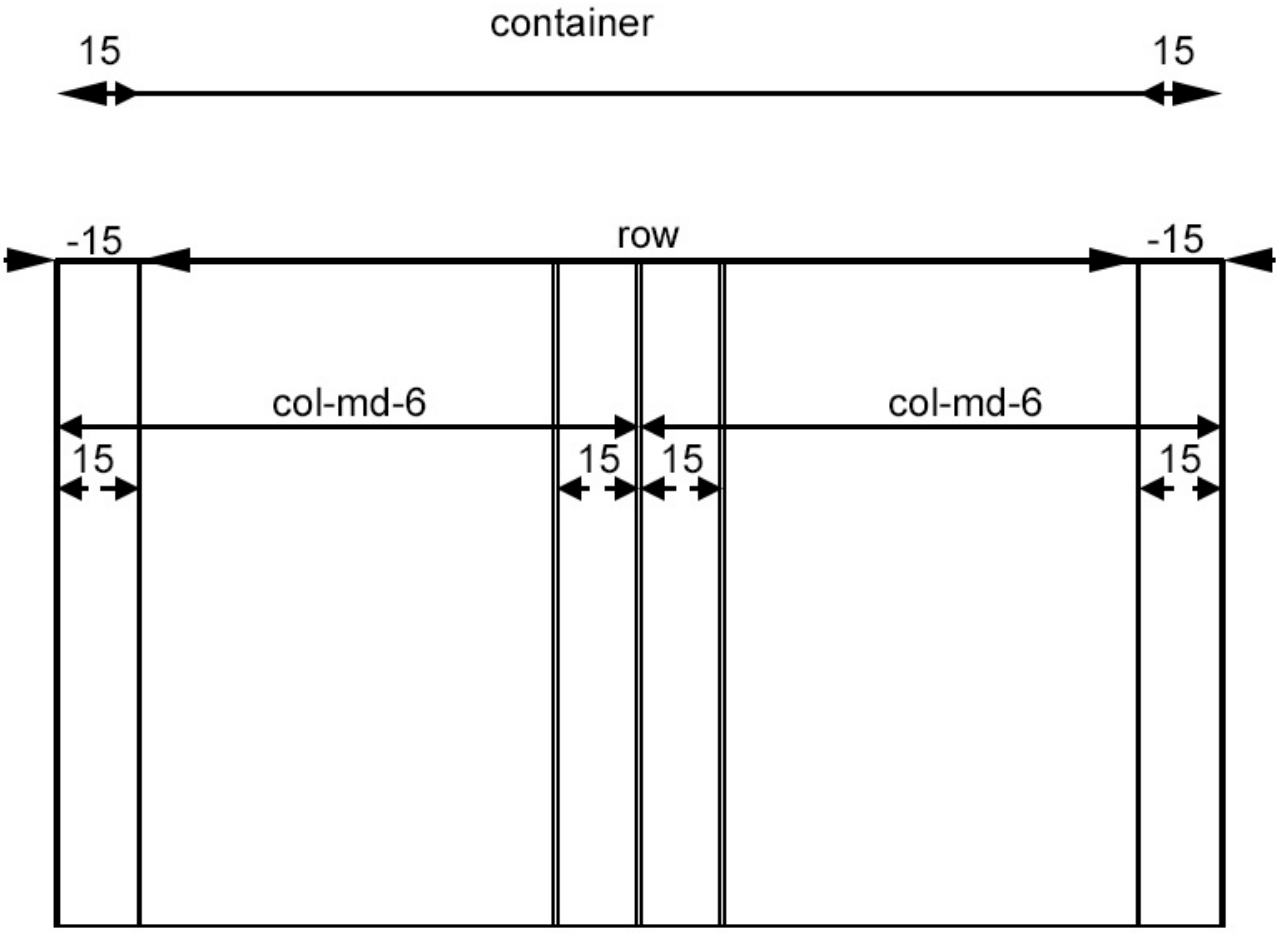


Рис. 17. Распределение классов в макете Bootstrap

Параметры и классы для адаптивности представлены в таблице 18.

Таблица 18. Настройка сетки

	Совсем небольшие устройства (<768px)	Небольшие устройства (>=768px)	Средние устройства (>=992px)	Большие устройства (>=1200px)
Ширина контейнера	None (auto)	750px	970px	1170px
Префикс класса	.col-xs-	.col-sm-	.col-md-	.col-lg-
Кол-во колонок	12			
Ширина колонки	Auto	~62px	~81px	~97px
Ширина зазора	30px (15px с каждой стороны колонки)			
Вложение	Yes			
Смещение	Yes			
Упорядочение колонки	Yes			

КонтентBootstrap [32]

Bootstrap содержит простые и легко настраиваемые шрифты для заголовков, основного текста, списков и многое другое.

Bootstrap устанавливает базовые глобальные отображения, типография, и стили ссылок. В частности, можно:

- Использовать **$body-bg** для того чтобы задать **background-color** в <body> (#fff по умолчанию)
- Использовать **$font-family-base**, **$font-size-base** и **$line-height-base** атрибуты в типографике.
- Установить глобальный цвет ссылки при помощи **$link-color** и применить подчеркивание ссылки только при наведении **:hover**.

Все HTML заголовки, от<h1> – <h6>, являются доступными. Также доступны классы от.h1 – .h6, в том случае, если необходимо изменить стиль заголовков.

Простое создание, вторичного текста (рис. 18) в любом заголовке осуществляется при помощи тега <small> или класса .small.

```
<h3>
  Забавный заголовок
  <small class="text-muted">с дополнительным текстом</small>
</h3>
```

Забавный заголовок с дополнительным текстом

Рис. 18. Вторичный заголовок

Для работы с изображениями предусмотрены различные классы, чтобы изменить их стиль и сделать адаптивными. Для адаптивности используется класс **.img-fluid**. Это позволяет менять размеры изображения вместе в родительским элементом.

```
<img src="..." class="img-fluid" alt="Responsive image">
```

Для изменения стилей изображения (рис. 19) добавляются следующие классы

```
<img src="..." alt="..." class="img-rounded">
<img src="..." alt="..." class="img-circle">
<img src="..." alt="..." class="img-thumbnail">
```

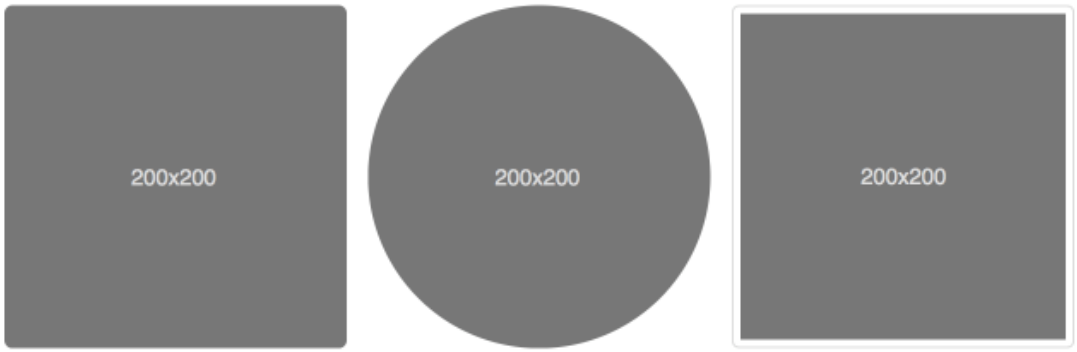


Рис. 19. Изменение стиля изображений

Для форматирования таблиц добавляется просто класс **.table** для любого контейнера **<table>**.

Простой пример таблицы Bootstrap (рис. 20).

#	Имя	Фамилия	Ник
1	Марк	Отто	@mdo
2	Джейкоб	Тортон	@fat
3	Ларри	the Bird	@twitter

Рис. 20. Пример форматирования таблицы

Для форматирования заголовков таблицы используется тег **<thead>** с классом **.thead-inverse**.

```
<table class="table">
  <thead class="thead-inverse">
    <tr>
```

Для чередования серых и белых строк используется класс **.table-striped** для тега **<table>**.

```
<table class="table table-striped">
  <thead>
    <tr>
```

С помощью класса **.table-hover** можно реализовать событие при наведении на строчки в пределах **<tbody>** .

Для создания стильных кнопок можно использовать любой из семи классов (рис. 21).

Рис. 21. Классы для стилизации кнопки

Код выглядит следующим образом:

```
<button type="button"
        class="btn btn-primary">Primary</button>
```

Полную информацию по компонентам можно найти на официальном сайте разработчиков https://getbootstrap.com/docs/4.0/components/alerts.

Проект Bootstrap динамично развивается. Разработчики стараются учитывать все тенденции Web-стандартов, поэтому новые функции и возможности появляются очень быстро. Уже сейчас Bootstrap поддерживается большинством CMS, сред разработки, а система Joomla сделала его основой своего проекта.

2. ПРАКТИЧЕСКИЙ РАЗДЕЛ

2.1. План практических работ

Практическая работа 1.Основы создания Web-документов

Цель работы: научиться создавать Web-страницы, оформлять внешние и внутренние ссылки, работать с изображениями средствами языка гипертекстовой разметки HTML.

Вопросы, изучаемые на лабораторной работе

Создание Web-страницы.

Работа с изображениями.

Установка ссылок.

Выравнивание текста и задание цвета.

Контрольные вопросы

Что представляют собой HTML-страницы? Каковы принципы гиппертекстовой разметки? Как выглядит схема построения Web-страниц? Что такое HTML-тег? Какие теги обеспечивает вставку изображений на Web-страницы?

Практическая работа 2. Форматирование текста

Цель работы: познакомимся с возможностью форматировать текст средствами языка гиппертекстовой разметки HTML.

Вопросы, изучаемые на лабораторной работе

Подключение CSSдля редактирования текста.

Работа с тегом pre.

Работа с надстрочными символами.

Создание списков.

Контрольные вопросы

Как создать нумерованный список на Web-странице? Как можно ввести формулу на Web-страницу. Как изменить цвет и размер шрифта выделенного фрагмента?

Практическая работа 3. Создание таблиц в HTML.

Цель работы: на практических примерах научиться создавать и форматировать таблицы с помощью языка HTML.

Вопросы, изучаемые на лабораторной работе

Работа с таблицами.

Форматирование ячеек.

Объединение ячеек и строк.

Чередование строк.

Контрольные вопросы

Какой тег определяет таблицу? Как определяется строка таблицы? Как определить заголовок таблицы? Как задать ячейку таблицы? Как задать цвет ячейки таблицы?

Практическая работа 4. Табличное представление сайта.

Цель работы: по макету создать две Web-страницы методом табличной верстки и научиться создавать ссылки на изображения.

Вопросы, изучаемые на лабораторной работе

Табличная верстка.

Задание фиксированной ширины таблицы.

Создание меню в виде таблицы.

Контрольные вопросы

Принцип табличной верстки? Как создать макет Web-страницы с фиксированной шириной? Как сделать ссылку в меню?

Практическая работа 5. Работа с формами

Цель работы: познакомиться с возможностями Web-форм, изучить элементы Web-форм и способах их форматирования. На занятии научимся создавать форму для отправки сообщения и способах изменения ее внешнего вида.

Вопросы, изучаемые на лабораторной работе

Web-форма.

Поле ввода текстовой информации.

Элементы переключения (чекбоксы и радиокнопки).

Создание кнопок.

Форматирование Web-формы.

Контрольные вопросы

Какой тег задается формы? Как определяются элементы управления формы? Как создать элемент формы типа радио-кнопка? Как создать текстовое поле в форме?

Практическая работа 6. Форматирование изображений с помощью CSS

Цель работы: научиться добавлять эффекты для изображений на Web-странице и создавать мини галереи, используя готовые библиотеки скриптов JavaScript.

Вопросы, изучаемые на лабораторной работе

Создание галереи изображений.

Подключение скриптов JavaScript.

Работа с внешними плагинами, настройка.

Контрольные вопросы

Зачем нужны каскадные таблицы стилей? Что такое селектор? Как задается цвет и фон элементов? Как форматировать внешний вид изображений с помощью CSS?

Практическая работа 7. Видео-контент и работа с фреймами

Цель работы: изучить теги HTML для добавления видео на Web-страницу, встраивать внешние Web-страницы через фреймы и строить фреймовые структуры.

Вопросы, изучаемые на лабораторной работе

Создание плавающего фрейма.

Деление страницы на фреймы.

Управление отображением Web-страницы.

Контрольные вопросы

Что такое фрейм? Как создать страницу с фреймами? Как добавить видео на страницу? Сколько фреймов может быть на странице?

Практическая работа 8. Анимация в CSS

Цель работы: изучить основы создания анимации в CSS, познакомиться с ключевой анимацией и научиться вращать и перемещать элементы Web-страницы HTML.

Вопросы, изучаемые на лабораторной работе

Создание ключевой анимации.

Анимация вращения, перемещения и масштабирования.

Настройка свойств анимации.

Контрольные вопросы

Как организовать визуальные эффекты с помощью CSS? Как создать анимацию вращения. Как ускорить анимацию в CSS? Как прервать анимацию при наведении?

Практическая работа 9. Блочная структура CSS. Создание меню

Цель работы: изучить блочную верстку средствами языка гипертекстовой разметки HTML и стилевой разметки CSS. Научиться выравнивать блоки div и создавать меню из списков.

Вопросы, изучаемые на лабораторной работе

Создание блочной структуры.

Понятие блоков в CSS.

Горизонтальное расположение блоков.

Свойство float.

Контрольные вопросы

Какие виды позиционирования блоковых элементов предусмотрены в CSS? Как создать горизонтальное меню? Как влияет параметр float на позицию блоков? Как задать фиксированную ширину для блоков?

Практическая работа 10. Работа с макетом Bootstrap

Цель работы: познакомиться с фреймворком Bootstrap, научиться создавать макеты для верстки, используя фреймворк Bootstrap, и создать страницу с меню и основным содержанием.

Вопросы, изучаемые на лабораторной работе

Понятие фреймворка.

Подключение фреймворка Bootstrap.

Создание макета Bootstrap.

Меню в Bootstrap.

Контрольные вопросы

Что такое фреймворк? Как подключить фреймворк Bootstrap на Web-страницу? Как создать меню в Bootstrap?

Практическая работа 11. Структура страницы для разных разрешений мониторов в Bootstrap

Цель работы: научиться адаптировать Web-страницы под разные разрешения мониторов и мобильных устройств, создать шаблоны с различными расположениями основных блоков на Web-странице.

Вопросы, изучаемые на лабораторной работе

Адаптивность в фреймворке Bootstrap.

Блочное распределение для мобильной версии сайта.

Блочное распределение для планшетной версии сайта.

Контрольные вопросы

Как создать макет для мобильной версии Web-сайта? Как изменить расположение основных блоков сайта для широкоформатной версии Web-сайта?

Практическая работа 12. Создание лэндинг страницы в Bootstrap

Цель работы: познакомиться с принципом построения лэндинг сайта с помощью HTML5 и фрэймворка Bootstrap, на примере шаблона самостоятельно построить лэндинг сайта.

Вопросы, изучаемые на лабораторной работе

Лэндинг страница.

Форматирование меню Bootstrap.

Форматирование изображений и текста с помощью стилей Bootstrap.

Контрольные вопросы

Как создать круглые рамки вокруг изображения? Как изменить цвет фона меню Bootstrap? Как прикрепить подвал сайта к нижней части страницы?

Практическая работа 13. Работа с формами в Bootstrap

Цель работы: научиться работать с форматированием форм с помощью фреймворка Bootstrap, добавлять интерактивные элементы на Web-страницу с табличным и строчным расположением

Вопросы, изучаемые на лабораторной работе

Создание форм в Bootstrap.

Основные элементы формы.

Форматирование элементов Web-форм?

Контрольные вопросы

Как форматировать радиокнопки в Bootstrap? Как создать выпадающий список в Bootstrap? Как с помощью Bootstrap расположить элементы формы в один ряд?

Практическая работа 14. Работа со слайдшоу и иконками в Bootstrap

Цель работы: научиться добавлять на Web-страницу галерею изображений с помощью фреймворка Bootstrap и использовать дополнительные эффекты форматирования..

Вопросы, изучаемые на лабораторной работе

Создание галереи изображений в Boostrap.

Созданий слайдера изображений.

Вставка иконок и выделение фрагментов текста.

Контрольные вопросы

Как создать слайдер изображений в Bootstrap? Какие компоненты Bootstrap можно встраивать на Web-страницу?

3. РАЗДЕЛ КОНТРОЛЯ ЗНАНИЙ

3.1. Вопросы для подготовки к зачетам и экзаменам

3.1.1. Вопросы для подготовки к зачету (третий семестр)

Понятие о доменном имени.

Публикация сайта в сети Интернет.

Этапы разработки сайта.

Виды сайтов.

Правила создания прототипа сайта. Схема и структура.

Правила адаптивного дизайна сайта.

Наполнение сайта.

Структура HTML документа.

Работа с абзацами, форматирование текста.

Ссылки в HTML: внешние ссылки

Ссылки в HTML: локальные ссылки

Встраивание графических элементов на страницу HTML.

Элемент BODY, изображение и цвет.

Таблицы. Атрибуты тега <TABLE>

Таблицы. Атрибуты тега <TR>

Таблицы. Атрибуты тегов <TD>, <TH>

Табличная верстка сайта.

Формы: простое и многострочное текстовые поля, поле для паролей.

Формы: списки, кнопки-переключатели.

Формы создание кнопок.

Создание стилей для элементов форм.

Способы использования CSS в HTML-документах.

Свойства CSS: шрифт

Свойства CSS: цвет и фон.

Свойства CSS: текст.

Применение эффектов к изображениям.

Строчные и блочные элементы.

Создание блока div.

Форматирование границ блочных элементов в CSS.

Подключение внешних библиотек JavaScript.

Создание плавающего фрейма.

Построение фреймовой структуры HTML-страницы.

Понятие анимации в CSS.

Создание ключевых кадров.

Работа со свойством transform.

Подключение библиотеки animate.css.

4.1.2. Вопросы для подготовки к экзамену (четвертый семестр)

Элементы уровня текста: элементы физического форматирования.

Элементы уровня текста: логические элементы.

Символьные примитивы.

Ссылки в HTML: внешние ссылки

Ссылки в HTML: внутренние ссылки

Ссылки в HTML: локальные ссылки

Работа с рисунками в HTML.

Элемент BODY, изображение и цвет.

Шрифтовое оформление: размер шрифта, начертание шрифта, выбор шрифта по его названию.

Таблицы. Атрибуты тега <TABLE>

Таблицы. Атрибуты тега <TR>

Таблицы. Атрибуты тегов <TD>, <TH>

Способы включения CSS в HTML-документы.

Классы и идентификаторы.

Свойства CSS: шрифт.

Свойства CSS: цвет и фон.

Свойства CSS: текст.

Блоки в HTML.

Виды блоков.

Ширина и высота блока.

Параметры обтекания блоков.

Работа со списками.

Работа с фрэймворком Bootstrap.

Подключение библиотеки Bootstrap.

Правила создания макета, используя Bootstrap.

Верстка основных блоков.

Элементы меню с помощью CSS.

Закрепление блоков.

Основные компоненты фрэймворка Bootstrap.

Работа с изображениями.

Адаптивная верстка.

Верстка лэндинг страницы.

Форматирование заголовков.

Работа с формами

Поля ввода.

Множественный выбор

Радио кнопки.

Выпадающий список

Создание слайдера

Работа с иконками

Преимущества работы с фрэймворком Bootstrap

4. ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

4.1. Учебная программа

ЧАСТНОЕ УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
"ИНСТИТУТ СОВРЕМЕННЫХ ЗНАНИЙ ИМЕНИ А.М. ШИРОКОВА"

УТВЕРЖДАЮ
Ректор Института современных знаний
имени А.М.Широкова
_____ А.Л. Капилов
_____
(дата утверждения)
Регистрационный № УД-_____/уч.

ОСНОВЫ WEB-ДИЗАЙНА
Учебная программа учреждения высшего образования
по учебной дисциплине для специальности:
1-19 01 01 Дизайн (по направлениям), направление специальности
1-19 01 01- 06 Дизайн (виртуальной среды)

2016 г.

Учебная программа составлена на основе образовательного стандарта ОСВО 1-19 01 01-2013 и учебного плана Института современных знаний имени А.М. Широкова по направлению специальности 1-19 01 01 - 06 Дизайн (виртуальной среды)

СОСТАВИТЕЛЬ:
Ю.Д. Васильева, доцент кафедры высшей математики и информатики Института современных знаний имени А.М. Широкова, кандидат технических наук

РЕЦЕНЗЕНТЫ:
кафедра дизайна Института современных знаний имени А.М. Широкова;
Ю.В. Виланский, ведущий инженер-программист Научно-производственного частного унитарного предприятия «Тетраэдр», кандидат технических наук, доцент

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:
Кафедрой высшей математики и информатики Института современных знаний имени А.М.Широкова
(протокол № 9 от 11 мая 2016 года);
Научно-методическим советом Института современных знаний имени А.М.Широкова
(протокол № 4 от 30 июня2016 года)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

С развитием Интернет технологий Web-ресурсы представляют собой набор инструментов с интерактивными возможностями, повышенной системой безопасности и упрощенной технологией построения Web-страниц. В среде Web-дизайна постоянно появляются новые языки Web-разработки, которые упрощают и предоставляют возможность создать полноценную Web-страницу в сети Интернет с разным уровнем знаний. Знание основ Web-дизайна позволит понять специфику создания сайтов разной степени сложности и построить свою Web-страницу с небольшим набором функций.

Цель дисциплины – изучение основных тенденций и требований предъявляемых к дизайну Web-сайтов; раскрытие аспектов работы программной составляющей Web-сайта любой степени сложности; обеспечение понимания основных принципов построения работы над Web-проектом.

Для достижения этой цели необходимо решение следующих учебных задач:

ознакомление студентов с основными концепциями Web-технологий;

усвоение знаний о правилах построения Web-страниц;

овладение сетевыми технологиями и использованием глобальной сети Интернет и ее служб;

изучение языка гипертекстовой разметки;

изучение технологий и средств визуального оформления Web-ресуров;

овладение эффективными методами и средствами верстки Web-страниц.

Дисциплина «Web-дизайн» относится к циклу специальных дисциплин и опирается на знания и умения студентов, полученные при изучении таких дисциплин как «Основы информационных технологий» и «Информационные технологии в дизайне».

Освоение образовательной программы по учебной дисциплине «Основы Web-дизайна» должно обеспечить формирование следующих академических компетенций:

АК-1. Владеть базовыми научно-теоретическими знаниями в области художественных, научно-технических, общественных и практических задач профессиональной деятельности.

АК-2. Владеть методикой системного и сравнительного анализа, междисциплинарным подходом к решению проблем, находить решения на стыке разных дисциплин, связанных с теорией и практикой дизайна.

АК-3. Владеть исследовательскими навыками.

АК-4. Уметь работать самостоятельно.

АК-5. Быть способным к творческой, креативной работе.

АК-6. Владеть междисциплинарным подходом при решении проблем.

АК-7. Иметь навыки использования современ6ных технических средств обработки информации.

АК-9. Уметь учиться, быть расположенным к постоянному повышению профессиональной квалификации.

Также студент должен приобрести следующие социально-личностные компетенции:

СЛК-2. Совершенствовать и развивать свой интеллектуальный и общекультурный уровень, повышать проектно-художественное мастерство.

СЛК-6. Быть способным к критике и самокритике.

После изучения учебной дисциплины студент должен владеть следующими профессиональными компетенциями и быть способным:

ПК-2. Осуществлять дизайн-проектирование с учетом соотношения смыслообразующих и формообразующих факторов (художественно-формальных, эргономических, инженерно-психологических, технологических, конструктивных, экологических, социально-культурных, экономических) в условиях как аналогового, так и безаналогового проектирования.

ПК-3. Формировать выразительное образное решение объекта проектирования на основе конкретного содержания.

ПК-4. Осуществлять прогностическое дизайн-проектирование с использованием инновационных технологий.

ПК-5. Осуществлять экспертную оценку уровня дизайнерского решения по основным смыслообразующим и формообразующим факторам.

ПК-6. Адаптироваться к изменению объекта профессиональной деятельности, как в пределах специализации, так и направление специальности.

ПК-7. Осуществлять развитие научно-теоретической и практической базы обеспечения дизайн-деятельности.

ПК-9. Собирать, анализировать и систематизировать профессиональный опыт в области дизайн-деятельности.

ПК-10. Выявлять общие закономерности функционирования и развития дизайн-деятельности на основе собранного факто логического материала.

ПК-11. Анализировать композиционные, конструктивные, технологические, эргономические и колористические решения продуктов дизайн-деятельности.

ПК-12. Анализировать результаты собственных дизайн-решений.

ПК-18. Уметь проектировать, организовывать, анализировать процесс педагогического взаимодействия при освоении профессиональных компетенций по направлению специальности.

В результате изучения дисциплины студенты должны:

знать:

основные тенденции и требования предъявляемые к построению Web-сайтов;

аспекты работы frontend-разработчиков.

основные принципы построения Web-сайтов.

уметь:

оценивать и анализировать технические задания на создание Web-сайта корпорации (предприятия, фирмы) и контент-проекта;

создавать Web-страницы и организовывать их в единую структуру Web-узла;

иметь представление:

о возможностях построения Web-сайтов любого уровня сложности;

о техническом и программном обеспечении Web-сайтов;

об основных принципах работы с системой управления контентом сайта.

В соответствии с учебными планами специальности «Дизайн» (по направлениям), направление специальности «Дизайн (виртуальной среды)» дисциплина изучается на протяжении двух семестров. Общее количество часов – 160, в том числе аудиторных – 86 часов. На самостоятельную работу отводится 74 часа.

Форма получения высшего образования – очная.

Для набора 2015 года дисциплина изучается в четвертом и пятом семестрах: 12 часов лекций и 34 часа практических занятий и 34 часа самостоятельной работы – в четвертом семестре и 12 часов лекций и 28 часов практических занятий и 40 часов самостоятельной работы – в пятом семестре. Текущая аттестация по дисциплине в четвертом семестре проводится в форме зачета, а в пятом семестре – в форме экзамена.

Начиная с набора 2016 года дисциплина изучается третьем и четвертом семестрах: 12 часов лекций и 34 часа практических занятий и 34 часа самостоятельной работы – в третьем семестре и 12 часов лекций и 28 часов практических занятий и 40 часов самостоятельной работы – в четвертом семестре. Текущая аттестация по дисциплине в третьем семестре проводится в форме зачета, а в четвертом семестре – в форме экзамена.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Тема 1. Основные понятия Web-дизайна

Определение понятия Web-сайта и Web-серверов, их цели и задачи. Визуальное представление Web-сайта как конечный продукт Web-дизайна. Понятие Web-разработки и основные требования, предъявляемые к специалистам данной области. Востребованность и перспективы работы в Web-сфере.

Направления и области использования глобальной сети. Интернет в Беларуси. Организация хранения и предоставления информации в Интернет.

Web-страница – логическая единица информации в Интернет. Определение понятия Всемирной паутины. Составляющие WWW: протоколы передачи данных HTTP, система адресации URI и язык гипертекстовой разметки HTML. Хранение Web-страниц и организация хостинг-компаний.

Создание доменного имени. Основные отличия и требования.

Наполнение сайта информацией. Оптимизация под поисковые системы. Отслеживание статистики посещения.

Тема 2. Базовые составляющие Web-дизайна

Использование цвета в Web-дизайне. Специфика представления цвета на экране монитора и его зрительное восприятие. Цветовые режимы, часто используемые при создании компьютерной графики. Способы записи цвета в HTML.

Понятие формы в Web-дизайне. Основные формы, используемые в сети интернет. Специфика зрительного восприятия каждой формы. Представление форм на Web-страницах и их запись в коде HTML.

Текст в Web-дизайне. Основные требования, предъявляемые к тексту в Интернет. Понятие структурирования текста. Основные и современные методы представления текста на Web-страницах и тенденции их развития.

Понятие размера графики и шрифта текста в Web-дизайне. Понятие взаиморасположения элементов на Web-странице.

Вертикальное и горизонтальное расположение информация. Представление информации на одной странице. Создание landing-страниц. Виды и специфика многостраничных сайтов.

Тема 3. Создание графики для Web-страниц

Основные форматы графических файлов. Основные графические решения, реализуемые программой Adobe Photoshop для сети интернет. Инструментарий Adobe Photoshop в контексте Web-разработок.

Оптимизация и сохранение изображений для дальнейшего использования в сети интернет.

Тема 4. Стандарт HTML. Основные концепции разметки

HTML-страницы. Принципы гипертекстовой разметки. Схема построения Web-страниц. Группы тегов HTML. Структура HTML-документа и элементы разметки заголовка документа. Контейнеры тела документа. Теги управления разметкой. Теги управления отображением символов. Гипертекстовые ссылки в HTML-документе.

Тема 5. Использование текстовых и графических элементов в HTML

Вставка изображений на Web-страницы и управление их отображением.

Создание таблиц. Теги разметки таблицы. Расположение таблицы на странице. Верстка документов с помощью таблиц.

Тема 6. Интерактивные элементы в HTML

Задание форм. Определение элементов управления формы. Создание многострочных областей ввода текста. Использование списков в форме. Методы форм.

Основные принципы работы фреймов. Создание простой страницы с фреймами. Вложенные и множественные кадровые структуры. Управление выводом во фреймах.

Тема 7. Каскадные таблицы стилей CSS

Назначение каскадных страниц стилей. Встраивание таблиц стилей в документ. Группирование и наследование. Селекторы. Псевдоклассы. Применение таблиц стилей. Модель форматирования. Встроенные элементы. Свойства форматирования элементов. Шрифты. Цвет и фон. Форматирование текста. Блоки. Визуальное форматирование. Визуальные эффекты.

Тема 8. Блочное представление элементов на странице

Контейнеры CSS. Различие между селекторами класса и id. Позиционирование элементов Web-страницы. Выравнивание и отступы.

Форматирование списков. Создание горизонтального и вертикального меню. Выпадающие элементы.

Скрытие и отображение элементов с помощью CSS.

Тема 9. Элементы анимации в HTML

Назначение JavaScript. Основные элементы. Типы данных и значения.

Использование готовых библиотек JavaScript. Создание галереи изображений, интерактивные формы. Всплывающие окна.

Тема 10. Построение Web-страниц с помощью фрэймворка Bootstrap

Понятие фрэймворка. Анализ существующих фрэймворков, их сравнение. Использование готовых шаблонов.

Создание макета сайта в редакторе Adobe Photoshop, используя сетку bootstrap. Редактирование элементов. Использование шрифта и иконок. Адаптивность Web-сайта к различным устройствам.

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов					Самостоятельная работа студентов (СРС)	Форма контроля знаний
		Лекции	Практические занятия	Семинарские занятия	Лабораторные занятия	Иное		
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	Основные понятия Web-дизайна	2					8	Результаты выполнения заданий по теме практического занятия
2	Базовые составляющие Web-дизайна							
3	Создание графики для Web-страниц	2	4					
4	Стандарт HTML. Основные концепции разметки	2	4				8	
5	Использование текстовых и графических элементов в HTML	2	8					
6	Интерактивные элементы в HTML	2	10				12	
7	Каскадные таблицы стилей CSS	2	8				6	
8	Блочное представление элементов на странице	4	8				6	Выполнение итогового самостоятельного задания
9	Элементы анимации в HTML	4	8				16	
10	Построение Web-страниц с помощью фрэймворка Bootstrap	4	12				18	
	Итого	**24**	**62**				**74**	

ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ
СПИСОК ЛИТЕРАТУРЫ

Основная

1. Веру Л. Секреты CSS. Идеальные решения ежедневных задач / Л. Веру. – СПб.: Питер Мейл, 2016. – 336 с.
2. Дакетт, Дж. HTML и CSS. Разработка и дизайн веб-сайтов / Джон Дакетт. – М.: Эксмо, 2013. – 480 с.
3. Джонсон, Дж. Web-дизайн: типичные ляпы и как их избежать / Дж. Джонсон. – М.: Кудиц-образ , 2005. – 400 с.
4. Дронов, В. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В. Дронов. – СПб.: БХВ-Петербург, 2014. – 416 с.
5. Дронов, В. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / В. Дронов, Н. Прохоренок. – СПб.: БХВ-Петербург, 2015. – 768 с.
6. Лебедев, Э.И. Web-дизайн с нуля! (+CD) / Э.И. Лебедев. – М.: Лучшие книги, 2008. – 336 с.
7. Левин, М.П. 2 в 1: самоучитель разработки Web-сайтов: HTML, CSS, графика, анимация, раскрутка (+ DVD) / М.П. Левин, Ю.М. Алексеев. – М.: Триумф, 2007. – 400 с.
8. Макфарленд, Д. Новая большая книга CSS / Д. Макфарленд. – СПб.: Питер Мейл, 2016. – 720 с.
9. Хатсон, Шерри. Photoshop для Web-дизайна / Шерри Хатсон. – М.: Кудиц-образ, 2006. – 240 с.
10. Холмогоров, В. Основы Web-мастерства: учебный курс / В. Холмогоров. – СПб.: Питер, 2002. – 350 с.
11. Хольцшлаг, Молли. 250 секретов HTML и WEB-дизайна / Молли Хольцшлаг. – М.: НТ Пресс, 2006. – 496 с.
12. Чебыкин, Р. Разработка и оформление текстового содержания сайта / Р. Чебыкин. – СПб.: БХВ-Петербург, 2004. – 528 с.

Дополнительная

1. Брезгунова, И. В. Основы веб-проектирования: учеб.-метод. пособие / И.В. Брезгунова, С.Н. Гринчук – Минск: РИВШ, 2013. – 126 с.
2. Велихов, С. Справочник по HTML 4.0 / С. Велихов. – М.: Бук-пресс, 2006. – 412 с.
3. Дронов, В.А. Самоучитель Adobe Dreamweaver CS5.5 / В.А. Дронов – СПб.: БХВ-Петербург, 2012. – 384 с.
4. Едомский, Ю. Техника Web-дизайна для студента / Ю. Едомский. – СПб.: BHV, 2012. – 400 с.
5. Мединов, О. Dreamweaver. Мультимедийный курс (+CD) / О. Мединов – СПб.: Питер, 2009. – 176 с.

6. Назарова, Ю. Компьютерная графика и Web-дизайн. Практикум по информатике (+ CD-ROM) / Ю. Назарова, Т. Немцова. – Форум, 2010.
7. Николаенко, Д.В. Практические занятия по JavaScript / Д.В. Николаенко. – М.: Наука и техника, 2000. – 128 с.
8. Ноблес, Р. Эффективный Web-сайт: учебное пособие / Р. Ноблес, К.Л. Греди. – М.: ТРИУМФ, 2004. – 560 с.
9. Петюшкин, А.В. HTML в Web-дизайне / А.В. Петюшкин – СПб.: БХВ-Петербург, 2004. – 400 с.
10. Полонская, Е.Л. Язык HTML. Самоучитель. / Е.Л. Полонская. – М.: Издательский дом Вильямс, 2003. – 203 с.
11. Снелл, Нэд. Абсолютно ясно о создании Web-страниц и Web-сайтов в цвете! Microsoft FrontPage / Нэд Снелл. – Москва: Триумф, 2005. – 217 с.
12. Сырых, Ю. Современный веб-дизайн. Эпоха Веб 3.0 / Юлия Сырых. – М.: Диалектика, 2012. – 368 с.
13. Сьярто, Джефф. Изучаем веб-дизайн / Д. Сьярто, Э. Уотролл. – М.: ЭКСМО, 2010.
14. Ташков, В.А. Веб-мастеринг на 100%: HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка / В.А. Ташков – СПб.: Питер, 2010. – 512 с.
15. Флэнаган, Д. JavaScript. Подробное руководство / Д. Флэнаган – Пер. с англ. – СПб: Символ-Плюс, 2008. – 992 с.

МЕТОДИЧЕСКИЕ ПОСОБИЯ

16. Виланский, Ю.В. Основы Web-дизайна: электронный курс лекций для студентов направления специальности "Дизайн (виртуальной среды)" / Ю.В. Виланский, Ю.Д. Васильева. – Минск: Институт современных знаний имени А.М.Широкова, 2012.

ЭЛЕКТРОННЫЕ РЕСУРСЫ

17. World Wide Web Consortium (W3C) [Электронный ресурс]. – Режим доступа: http://www.w3.org/ – Дата доступа: 02.05.2016
18. Все самое интересное о веб-дизайне [Электронный ресурс] – Режим доступа: http://www.designonstop.com – Дата доступа: 05.05.2016.
19. Портал вебмастеров [Электронный ресурс] – Режим доступа: http://www.webmasters.by/ – Дата доступа: 05.05.2016
20. Сайт о сайтах или как самостоятельно освоить основы WEB программирования и WEB дизайна [Электронный ресурс] – Режим доступа: http://sevidi.narod.ru/index.html/ – Дата доступа: 05.05.2016
21. Школа создания сайтов в Adobe Dreamweaver [Электронный ресурс] – Режим доступа: http://dreamweaver-school.narod.ru/ – Дата доступа: 05.05.2016

ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

1. Основы создания Web-документов.
2. Форматирование текста.
3. Создание таблиц в HTML.
4. Табличное представление сайта.
5. Работа с формами.
6. Форматирование изображений с помощью CSS.
7. Видео-контент и работа с фреймами.
8. Анимация в CSS.
9. Блочная структура CSS. Создание меню.
10. Работа с макетом Bootstrap.
11. Структура страницы для разных разрешений мониторов в Bootstrap.
12. Создание лэндинг страницы в Bootstrap.
13. Работа с формами в Bootstrap.
14. Работа со слайдшоу и иконками в Bootstrap.

ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

№ п/п	Название раздела, темы	Кол-во часов на СРС	Задание	Форма выполнения	Цель или задача СРС
1	**2**	**3**	**4**	**5**	**6**
1	Основные понятия Web-дизайна	8	Изучить литературные источники и электронные ресурсы по темам 1-3.	Чтение литературы, посещение интернет-ресурсов	Ознакомиться с основными возможностями и методами построения Web-проектов. Изучить требования к графике для Web-страниц.
2	Базовые составляющие Web-дизайна				
3	Создание графики для Web-страниц				
4	Стандарт HTML. Основные концепции разметки.	8	По заданному шаблону создать HTML страницу согласно заданиям для самостоятельной работы	Файлы в формате html	Закрепление навыков использования текстовых и графических элементов на Web-странице
5	Использование текстовых и графических элементов в HTML				
6	Интерактивные элементы в HTML	12	Используя библиотеки создать Web-галерею с интерактивными элементами	Файлы в формате html	Закрепление навыков использования интерактивных элементов в HTML
7	Каскадные таблицы стилей CSS	6	Построить Web-страницу, используя CSS стили	Файл в формате html	Закрепление навыков использования каскадных таблиц стилей

1	2	3	4	5	6
8	Блочное представление элементов на странице	6	Построить Web-страницу, используя блочное представление страниц	Файл в формате html	Закрепление навыков блочного представления страниц
9	Элементы анимации в HTML	16	Построить Web-страницу, содержащую элементы анимации	Файл в формате html	Закрепление навыков использования элементов анимации в HTML
10	Построение Web-страниц с помощью фрэймворка Bootstrap	18	Построить макет Web-страницы, используя фреймворк Bootstrap	Файлы в формате html	Закрепление навыков применения для верстки Web-страниц фреймворка Bootstrap

ПРОТОКОЛ СОГЛАСОВАНИЯ УЧЕБНОЙ ПРОГРАММЫ УВО

Название учебной дисциплины, с которой требуется согласование	Название кафедры	Предложения об изменениях в содержании учебной программы Института по учебной дисциплине	Решение, принятое кафедрой, разработавшей учебную программу (с указанием даты и номера протокола)

ДОПОЛНЕНИЯ И ИЗМЕНЕНИЯ К УЧЕБНОЙ ПРОГРАММЕ УВО

на _____ /_____ учебный год

№ пп	Дополнения и изменения	Основание

Учебная программа пересмотрена и одобрена на заседании кафедры высшей математики и информатики (протокол № ____от _____201__г.)

Заведующий кафедрой

_____ _____ _____

(ученая степень, ученое звание) (подпись) (И.О.Фамилия)

УТВЕРЖДАЮ
Декан факультета

_____ _____ _____

(ученая степень, ученое звание) (подпись) (И.О.Фамилия)

4.2. Основная литература

1. Веру Л. Секреты CSS. Идеальные решения ежедневных задач / Л. Веру. – СПб. : Питер Мейл, 2016. – 336 с.

2. Виланский, Ю. В. Основы Web-дизайна : электронный курс лекций для студентов направления специальности Дизайн (виртуальной среды) / Ю. В. Виланский, Ю. Д. Васильева. – Минск : Институт современных знаний имени А. М. Широкова, 2012.

3. Дакетт, Дж. HTML и CSS. Разработка и дизайн веб-сайтов / Джон Дакетт. – М. : Эксмо, 2013. – 480 с.

4. Джонсон, Дж. Web-дизайн: типичные ляпы и как их избежать / Дж. Джонсон. – М. : Кудиц-образ , 2005. – 400 с.

5. Дронов, В. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В. Дронов. – СПб. : БХВ-Петербург, 2014. – 416 с.

6. Дронов, В. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / В. Дронов, Н. Прохоренок. – СПб. : БХВ-Петербург, 2015. – 768 с.

7. Лебедев, Э. И. Web-дизайн с нуля! (+CD) / Э. И. Лебедев. – М. : Лучшие книги, 2008. – 336 с.

8. Левин, М. П. 2 в 1: самоучитель разработки Web-сайтов: HTML, CSS, графика, анимация, раскрутка (+DVD) / М. П. Левин, Ю. М. Алексеев. – М. : Триумф, 2007. – 400 с.

9. Макфарленд, Д. Новая большая книга CSS / Д. Макфарленд. – СПб. : Питер Мейл, 2016. – 720 с.

10. Хатсон, Шерри. Photoshop для Web-дизайна / Шерри Хатсон. – М. : Кудиц-образ, 2006. – 240 с.

11. Холмогоров, В. Основы Web-мастерства: учебный курс / В. Холмогоров. – СПб. : Питер, 2002. – 350 с.

12. Хольцшлаг, Молли. 250 секретов HTML и WEB-дизайна / Молли Хольцшлаг. – М. : НТ Пресс, 2006. – 496 с.

13. Чебыкин, Р. Разработка и оформление текстового содержания сайта / Р. Чебыкин. – СПб. : БХВ-Петербург, 2004. – 528 с.

4.3. Дополнительная литература

14. Брезгунова, И. В. Основы веб-проектирования : учеб.-метод. пособие / И. В. Брезгунова, С. Н. Гринчук – Минск: РИВШ, 2013. – 126 с.

15. Едомский, Ю. Техника Web-дизайна для студента / Ю. Едомский. – СПб. : BHV, 2012. – 400 с.

16. Мединов, О. Dreamweaver. Мультимедийный курс (+CD) / О. Мединов – СПб. : Питер, 2009. – 176 с.

17. Назарова, Ю. Компьютерная графика и Web-дизайн. Практикум по информатике (+ CD-ROM) / Ю. Назарова, Т. Немцова. – Форум, 2010.

18. Николаенко, Д. В. Практические занятия по JavaScript / Д. В. Николаенко. – М. : Наука и техника, 2000. – 128 с.

19. Ноблес, Р. Эффективный Web-сайт: учебное пособие / Р. Ноблес, К. Л. Греди. – М. : ТРИУМФ, 2004. – 560 с.

20. Петюшкин, А. В. HTML в Web-дизайне / А. В. Петюшкин – СПб. : БХВ-Петербург, 2004. – 400 с.

21. Полонская, Е. Л. Язык HTML. Самоучитель. / Е. Л. Полонская. – М. : Издательский дом Вильямс, 2003. – 203 с.

22. Снелл, Нэд. Абсолютно ясно о создании Web-страниц и Web-сайтов в цвете! Microsoft FrontPage / Нэд Снелл. – М. : Триумф, 2005. – 217 с.

23. Сьярто, Джефф. Изучаем веб-дизайн / Д. Сьярто, Э. Уотролл. – М. : ЭКСМО, 2010.

24. Ташков, В. А. Веб-мастеринг на 100%: HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка / В. А. Ташков – СПб. : Питер, 2010. – 512 с.

25. Флэнаган, Д. JavaScript. Подробное руководство / Д. Флэнаган – Пер. с англ. – СПб. : Символ-Плюс, 2008. – 992 с.

26. Jonston, Ollie& The Illusion of Life: Disney Animation / Ollie Jonston, Frank Thomas – Italy : Frank, 1981. – 576 p.

27. Сырых, Ю. Современный веб-дизайн. Эпоха Веб 3.0 / Ю. Сырых. – М. : Диалектика, 2012. – 368 с.

28. Велихов, С. Справочник по HTML 4.0 / С. Велихов. – М. : Бук-пресс, 2006. – 412 с.

29. Дронов, В. А. Самоучитель Adobe Dreamweaver CS5.5 / В. А. Дронов – СПб. : БХВ-Петербург, 2012. – 384 с.

4.4. Ресурсы Интернет

30. Блочная верстка или основы анатомии скелета сайтов [Электронный ресурс] – Режим доступа: http://www.internet-technologies.ru/articles/blochnaya-verstka-ili-osnovy-anatomii-skeleta-saytov.html – Дата доступа: 07.12.2017.

31. Все самое интересное о веб-дизайне [Электронный ресурс] – Режим доступа: http://www.designonstop.com – Дата доступа: 05.05.2016.

32. Контент в Bootstrap 4 [Электронный ресурс] – Режим доступа: http://bootstrap-v4.ru/content/reboot/ – Дата доступа: 26.01.2018.

33. Настройка оптимальной функциональности Photoshop [Электронный ресурс] – Режим доступа: https://webformula.pro/article/nastroyka-optimalnoy-funktsionalnosti-photoshop/ – Дата доступа: 07.11.2017.

34. Портал вебмастеров [Электронный ресурс] – Режим доступа: http://www.webmasters.by/ – Дата доступа: 05.05.2016.

35. Принципы анимации для веба [Электронный ресурс] – Режим доступа: https://habrahabr.ru/company/htmlacademy/blog/255583/ – Дата доступа: 12.01.2018.

36. Разметка Bootstrap 4 [Электронный ресурс] – Режим доступа: http://bootstrap-v4.ru/layout/overview/ – Дата доступа: 24.01.2018.

37. Сайт о сайтах или как самостоятельно освоить основы WEB программирования и WEB дизайна [Электронный ресурс] – Режим доступа: http://sevidi.narod.ru/index.html/ – Дата доступа: 05.05.2016.

38. Типографика в вебе [Электронный ресурс] – Режим доступа: https://habrahabr.ru/post/324944/ – Дата доступа: 27.09.2017.

39. Формы для сайта: особенности создания и оформления [Электронный ресурс] – Режим доступа: http://webstudio2u.net/ru/design-web/712-formy-dlya-saita.html – Дата доступа: 14.09.2017.

40. Цвет / Самоучитель HTML [Электронный ресурс] – Режим доступа: http://htmlbook.ru/samhtml/znacheniya-atributov-tegov/tsvet – Дата доступа: 05.08.2017.

41. Что такое лендинг [Электронный ресурс] – Режим доступа: https://habrahabr.ru/company/trinion/blog/273917 – Дата доступа: 18.10.2017.

42. Школа создания сайтов в Adobe Dreamweaver [Электронный ресурс] – Режим доступа: http://dreamweaver-school.narod.ru/ – Дата доступа: 05.05.2016.

43. Юзабилити веб-форм [Электронный ресурс] – Режим доступа: http://popel-studio.com/blog/article/usabiliti-web-form.html – Дата доступа: 14.09.2017.

44. Bootstrap 3 сетка: основы [Электронный ресурс] – Режим доступа: http://dnzl.ru/view_post.php?id=360 – Дата доступа: 25.01.2018.

45. CSS3-анимация [Электронный ресурс] – Режим доступа: https://html5book.ru/css3-animation – Дата доступа: 23.01.2018.

46. World Wide Web Consortium (W3C) [Электронный ресурс]. – Режим доступа: http://www.w3.org/ – Дата доступа: 02.05.2016.

4.5. Техническое и программное обеспечение дисциплины

Для проведения лекционных занятий используется аудитория, оснащенная персональным компьютером с подключенной к нему видеопроекционной установкой и экран, на который выводятся слайды презентаций, видеоролики и окна интерфейса программ по тематике лекций.

Для проведения лабораторных работ используется компьютерный класс с персональными компьютерами, работающими под управлением операционной системы Windows. Для выполнения лабораторных работ используется программы Brackets и Adobe Dreamweaver, бразуеры Chrome и Internet Explorer.

СОДЕРЖАНИЕ

Учебное электронное издание

Автор-составитель
Васильева Юлия Дмитриевна

ОСНОВЫ WEB-ДИЗАЙНА

*Электронный учебно-методический комплекс
для студентов специальности 1-19 01 01 Дизайн (по направлениям),
направление специальности 1-19 01 01-06 Дизайн (виртуальной среды)*

[Электронный ресурс]

Редактор *И. Б. Михнюк*
Технический редактор *Ю. В. Хадьков*

Подписано в печать 30.08.2018.
Гарнитура Times Roman. Объем 2,2 Мб

Частное учреждение образования
«Институт современных знаний имени А. М. Широкова»
Свидетельство о регистрации издателя №1/29 от 19.08.2013
220114, г. Минск, ул. Филимонова, 69.

ISBN 978-985-547-234-7